

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-191892

(43)Date of publication of application : 28.07.1995

(51)Int.Cl.

G06F 12/00

G06F 12/00

G11C 16/06

(21)Application number : 05-013980

(71)Applicant : MICROSOFT CORP

(22)Date of filing : 29.01.1993

(72)Inventor : KRUEGER WILLIAM J  
RAJAGOPALAN SRIRAM

(30)Priority

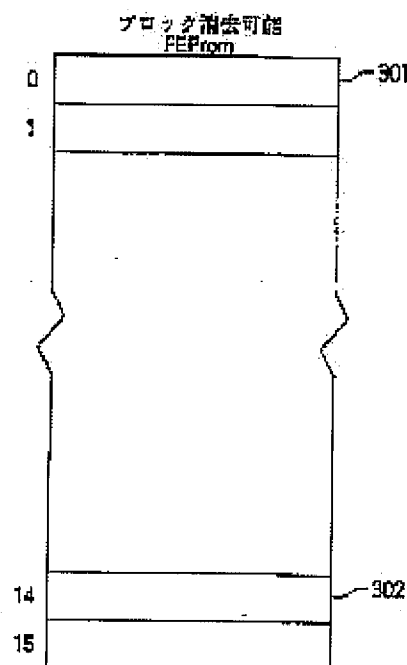
Priority number : 92 828763 Priority date : 29.01.1992 Priority country : US

(54) METHOD FOR MANAGING FILE SYSTEM BY USING FLASH ERASABLE  
PROGRAMMABLE READ ONLY MEMORY AND SYSTEM THEREFOR

(57)Abstract:

PURPOSE: To provide a method and system for managing a file stored in block erasable and flash erasable programmable read only memory.

CONSTITUTION: A manager for managing a file is provided with a block assigning routine for selecting a block 302 at a memory position for storing data, and each block 302 is provided with an assignment table and a data area divided into data areas, and the assignment table is provided with an entry corresponding to the data area. Moreover, the manager is provided with a data area assigning routine for selecting the data area in the data area for the selected block, selecting the entry of the assignment table so as to be made correspond to the data area, and setting the entry selected so as to be made correspond to the selected data area and the assigned state.



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平7-191892

(43) 公開日 平成7年(1995)7月28日

(51) Int.Cl. <sup>6</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 12/00	5 2 0 J	8944-5B		
	5 0 1 H	8944-5B		
G 1 1 C 16/06				
			G 1 1 C 17/ 00	5 3 0 B

審査請求 未請求 請求項の数41 O L (全 34 頁)

(21) 出願番号 特願平5-13980  
 (22) 出願日 平成5年(1993)1月29日  
 (31) 優先権主張番号 07/828763  
 (32) 優先日 1992年1月29日  
 (33) 優先権主張国 米国 (US)

(71) 出願人 391055933  
 マイクロソフト コーポレーション  
 MICROSOFT CORPORATI  
 ON  
 アメリカ合衆国 ワシントン州 98052-  
 6399 レッドモンド ワン マイクロソフ  
 ト ウェイ (番地なし)  
 (72) 発明者 ウィリアム ジェイ クルーガー  
 アメリカ合衆国 ワシントン州 98053  
 レッドモンド トゥーハンドレッドアンド  
 エイス アベニュー ノースイースト  
 6320  
 (74) 代理人 弁理士 中村 稔 (外6名)

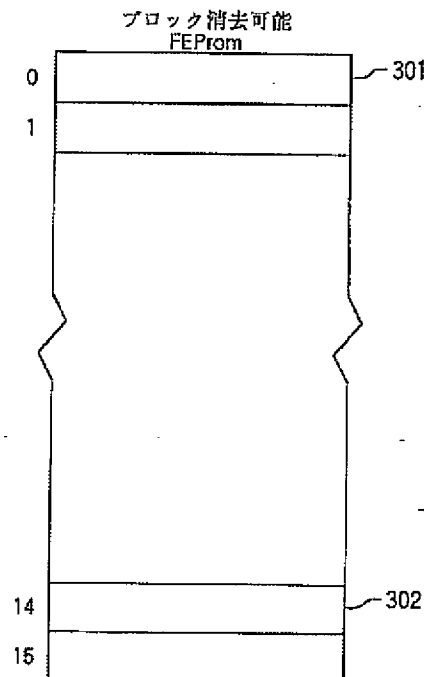
最終頁に続く

(54) 【発明の名称】 フラッシュ消去可能なプログラマブル・リードオンリメモリを用いてファイルシステムをマネー  
 ジする方法及びシステム

(57) 【要約】 (修正有)

【目的】 ブロック消去可能で且つフラッシュ消去可能  
 なプログラマブル・リードオンリメモリに記憶されたフ  
 ァイルをマネージする方法及びシステムを提供する。

【構成】 ファイルをマネージするためのマネージャ  
 は、データを記憶するメモリ位置のブロックを選択する  
 ためのブロック割り当てルーチンを備えており、各プロ  
 ックは、割り当てテーブルと、データエリアに分割され  
 たデータ領域とを有し、割り当てテーブルはデータエリ  
 アに対応するエントリを有している。更に、上記マネー  
 ジャは、上記選択されたブロックに対してデータ領域内  
 のデータエリアを選択し、それに対応するように割り当  
 てテーブルのエントリを選択し、そして選択されたデー  
 タエリア及び割り当てられた状態に対応するように上記  
 選択されたエントリをセットするためのデータエリア割  
 り当てルーチンを備えている。



(2)

特開平7-191892

1

## 【特許請求の範囲】

【請求項1】 コンピュータメモリをメモリ位置のブロックに分割し、各ブロックは、割り当てテーブルと、データエリアに分割されたデータ領域とを有するものであり、上記割り当てテーブルは上記領域のデータエリアに対応するエントリを有し、そしてデータを記憶するブロックを選択するためのブロック割り当てルーチンと、データを記憶する上記選択されたブロックに対しデータ領域内のデータエリアを選択し、その選択されたデータエリアに対応するように割り当てテーブルのエントリを選択し、そして上記選択されたデータエリア及び割り当てられた状態に対応するように上記選択された割り当てテーブルエントリをセットするためのデータエリア割り当てルーチンと、

上記選択されたデータエリアにデータを記憶するための記憶ルーチンとを備えたことを特徴とするコンピュータメモリ用のマネージャ。

【請求項2】 割り当てられた状態にある割り当てテーブルエントリを割り当て解除状態にセットするためのデータエリア割り当て解除ルーチンと、

その割り当て解除された状態にある割り当てテーブルエントリに対応するデータエリアをリクレーミングするためのブロックリクレーマルーチンとを更に備えた請求項1に記載のコンピュータメモリ用のマネージャ。

【請求項3】 各ブロックがヘッダ情報を有し、これらヘッダ及び割り当てテーブルから情報を収集しそしてその収集した情報をメモリキャッシュに記憶するための初期化ルーチンを更に備えた請求項1に記載のコンピュータメモリ用のマネージャ。

【請求項4】 1つ以上の論理ブロックを有する一回書き込み多数回読み取りのメモリデバイスにおいて、データを記憶するためのエリアに論理的に分割される各ブロック内のデータ領域と、各ブロック内に記憶されるブロック構造体であって、ヘッダと割り当てテーブルとを有して、ヘッダはブロックに特定の情報を含み、割り当てテーブルはデータ領域のエリアに対応するエントリを有していると共に、その対応するデータ領域のエリアに関連した情報を含んでいるようなブロック構造体とを備えたことを特徴とするメモリデバイス。

【請求項5】 上記ヘッダは、論理ブロック番号と、ブロック状態と、ブロック消去カウントとを含む請求項4に記載のメモリデバイス。

【請求項6】 割り当てテーブルエントリは、対応するデータ領域のエリアを指すポインタと、対応するデータ領域のエリアの長さを指示するレンジと、割り当てテーブルエントリのステータスとを含んでいる請求項4に記載のメモリデバイス。

【請求項7】 上記メモリデバイスは、ブロック消去可能なプログラマブル・リードオンリメモリである請求項

2

4に記載のメモリデバイス。

【請求項8】 複数のブロックを備えたブロック消去可能なプログラマブル・リードオンリメモリにおいてメモリをマネージする方法が、

各ブロックにブロックヘッダ情報を記憶し、

各ブロックに割り当てテーブルを記憶し、そして各ブロックのデータ記憶エリアにデータを記憶する、という段階を備えたことを特徴とする方法。

【請求項9】 上記ブロックヘッダ情報は、論理ブロック番号と、ブロック状態と、ブロック消去カウントとを含んでいる請求項8に記載の方法。

【請求項10】 上記割り当てテーブルはデータ記憶エリアの一部分に対応するエントリを含み、これらのエントリは、データ記憶エリアの対応する部分を指すポインタと、データ記憶エリアの対応部分の長さを指示するレンジと、エントリの状態を指示するステータスとを含む請求項8に記載の方法。

【請求項11】 ブロックを備えたブロック消去可能なプログラマブル・リードオンリメモリにおいて割り当て解除されたスペースをリクレーミングする方法が、

リクレーミングされるべきブロックにおいて割り当て解除された又は割り当てられたものとしてデータ領域を識別し、

スベアブロックを消去し、そしてリクレーミングされるべきブロックからスベアブロックへ割り当てられたデータ領域をコピーし、それにより、割り当て解除されたデータ領域に対応するメモリエリアを割り当てのためにリクレーミングする、という段階を備えたことを特徴とする方法。

【請求項12】 割り当てられたデータ領域はスベアブロック内の隣接するメモリ位置にコピーされる請求項11に記載の方法。

【請求項13】 コンピュータメモリデバイス内のデータ領域をアドレスする方法において、上記メモリはブロックに分割され、そして各ブロックは物理的なブロック番号を有しており、上記方法は、

各ブロックに割り当てテーブルを記憶し、この割り当てテーブルは、ブロック内のデータ領域のオフセットを指示するエントリと、エントリインデックスとを有しており、

更に、各ブロックに論理ブロック番号を記憶し、この論理ブロック番号と、割り当てテーブルのエントリインデックスとによってデータ領域を識別し、そして上記論理ブロック番号と、割り当てテーブルのエントリインデックスとに基づいてその識別されたデータ領域に対するアドレスを発生する、という段階を備えたことを特徴とする方法。

【請求項14】 アドレスを発生する上記段階は、上記論理ブロック番号から物理的なブロック番号を決定し、各ブロックは対応するスタートアドレスを有し、

(3)

特開平7-191892

3

上記割り当てテーブルのエントリインデックスによって指示された上記決定された物理的なブロック番号において上記割り当てテーブルのエントリからオフセットを検索し、そしてその検索されたオフセットを上記決定された物理的なブロック番号と共にブロックのスタートアドレスに追加して、識別されたデータ領域のアドレスを発生するという段階を含む請求項13に記載の方法。

【請求項15】 コンピュータメモリデバイスは、ブロック消去可能なプログラマブル・リードオンリメモリである請求項13又は14に記載の方法。

【請求項16】 コンピュータメモリデバイスのブロックを識別する方法において、各ブロックは物理的なブロック番号を有しており、上記方法は、各ブロックに論理ブロック番号を記憶し、各論理ブロック番号から、その論理ブロック番号が記憶された物理的なブロック番号へ至るマップを発生し、論理ブロック番号を受け取り、そしてその受け取った論理ブロック番号を、上記発生されたマップを用いて物理的なブロック番号に変換する、という段階を備えたことを特徴とする方法。

【請求項17】 ブロックをリクレーミングするときに、リクレーミングされるべきブロックからリクレーミングされたブロックへ論理ブロック番号をコピーする段階を備えた請求項16に記載の方法。

【請求項18】 コンピュータメモリデバイスは、ブロック消去可能なプログラマブル・リードオンリメモリである請求項16又は17に記載の方法。

【請求項19】 複数のブロックを有するブロック消去可能なプログラマブル・リードオンリメモリにおいてブロックの消去カウントを維持する方法が、ブロックを消去した回数を指示する消去カウントを各ブロックに記憶し、

第1ブロックを第2ブロックにコピーするときに、第2ブロックの消去カウントを保持し、そしてブロックを消去するときに、その消去の前に消去カウントを増加しそしてその増加した消去カウントを消去したブロックに記憶する、という段階を備えたことを特徴とする方法。

【請求項20】 データを記憶するためのブロック消去可能なプログラマブル・リードオンリメモリにおいてブロックを割り当てる方法が、

ブロックを消去した回数を示す消去カウントを各ブロックごとに維持し、

データを記憶するに十分なスペースを有するブロックを選択し、そしてデータを記憶するためにその消去カウントに基づいて上記選択されたブロックを識別し、ブロックの割り当てを行う、という段階を備えたことを特徴とする方法。

【請求項21】 ブロック消去可能なプログラマブル・リードオンリメモリにおいてブロックの消去を均等化する方法が、

4

消去された第1ブロックを識別し、

第1ブロックよりも少ない回数で消去された第2ブロックを識別し、そして第1ブロックのデータを第2ブロックのデータとスワッピングする、という段階を備えたことを特徴とする方法。

【請求項22】 複数のディレクトリエントリを有するハイアラキ式のディレクトリ構造体に基づいてファイルを編成するためのコンピュータファイル記憶システムにおいて、

10 ディレクトリエントリを記憶するメモリを有するコンピュータと、

ディレクトリエントリを記憶するメモリの一部分を割り当てる手段であって、ディレクトリエントリは、一次ポインタと、二次ポインタと、兄弟ポインタとを有し、これらのポインタが最初に所定値にセットされているような割り当て手段と、

ディレクトリエントリの上記兄弟ポインタを、ディレクトリ構造体の同じレベルにある別のディレクトリエントリを指すようにセットして、兄弟ディレクトリエントリのリンクされたリストを形成するための手段と、

20 ディレクトリエントリの上記一次ポインタを、ディレクトリ構造体の次に低いレベルにある別のディレクトリエントリを指すようにセットする手段と、

取って代えられるディレクトリエントリの上記二次ポインタを、取って代わるディレクトリエントリを指すようにセットする手段とを具備し、この取って代わるディレクトリエントリは、取って代えられるディレクトリエントリの更新されたデータを含むことを特徴とするコンピュータファイル記憶システム。

30 【請求項23】 上記取って代わるディレクトリエントリの一次ポインタを上記取って代えられるディレクトリエントリの一次ポインタに等しくセットするための手段と、

上記取って代わるディレクトリエントリの兄弟ポインタを上記取って代えられるディレクトリエントリの兄弟ポインタに等しくセットするための手段とを更に備えた請求項22に記載のコンピュータファイル記憶システム。

40 【請求項24】 ファイルエントリを記憶するためのメモリの一部分を割り当てる手段を更に備え、ファイルエントリは、関連ファイルに関する情報を含むものであり、そしてディレクトリエントリの一次ポインタをその割り当てられたファイルエントリを指すようにセットする手段を更に備えた請求項22に記載のコンピュータファイル記憶システム。

【請求項25】 上記の割り当てられたファイルエントリは、一次ポインタと、二次ポインタと、最初に所定値にセットされる兄弟ポインタとを含み、

ファイルエントリの上記兄弟ポインタを、別のファイル又はディレクトリエントリを指すようにセットして、フ

50 ァイル又はディレクトリエントリのリンクされたリスト

(4)

特開平7-191892

5

を形成するための手段と、

取って代えられるファイルエントリの上記二次ポイントを、取って代わるファイルエントリを指すようにセットする手段とを具備し、この取って代わるファイルエントリは、取って代えられるファイルエントリの更新されたデータを含む請求項24に記載のコンピュータファイル記憶システム。

【請求項26】 上記取って代わるファイルエントリの一次ポイントを上記取って代えられるファイルエントリの一次ポイントに等しくセットするための手段と、

上記取って代わるファイルエントリの兄弟ポイントを上記取って代えられるファイルエントリの兄弟ポイントに等しくセットするための手段とを更に備えた請求項25に記載のコンピュータファイル記憶システム。

【請求項27】 ファイル情報エントリを記憶するためのメモリの一部分を割り当てる手段を更に備え、ファイル情報エントリは、ファイルエクステンツに関する情報を含むものであり、そしてファイルエントリの一次ポイントをその割り当てられたファイル情報エントリを指すようにセットする手段を更に備えた請求項24に記載のコンピュータファイル記憶システム。

【請求項28】 上記ファイル情報エントリは、最初に所定値にセットされる一次ポイント及び二次ポイントを含み、

ファイル情報エントリの上記一次ポイントを、その同じファイルに関連した別のファイル情報エントリを指すようにセットして、その同じファイルに対するファイル情報エントリのリンクされたりストを形成するための手段と、

取って代えられるファイル情報エントリの上記二次ポイントを、取って代わるファイル情報エントリを指すようにセットする手段とを具備し、この取って代わるファイル情報エントリは、取って代えられるファイル情報エントリの更新されたデータを含む請求項27に記載のコンピュータファイル記憶システム。

【請求項29】 上記取って代わるファイル情報エントリの一次ポイントを上記取って代えられるファイル情報エントリの一次ポイントに等しくセットする手段を更に備えた請求項28に記載のコンピュータファイル記憶システム。

【請求項30】 ファイルを記憶するためのコンピュータファイル記憶システムにおいて、ファイルを記憶するメモリを有するコンピュータと、ファイルエントリを記憶するメモリの一部分を割り当てる手段であって、ファイルエントリは、それに関連したファイルに関する情報を含むと共に、一次ポイントと、二次ポイントと、最初に所定値にセットされる兄弟ポイントとを有しているような割り当て手段と、ファイルエントリの上記兄弟ポイントを、別のファイルエントリを指すようにセットして、ファイルエントリの

6

リンクされたりストを形成する手段と、

取って代えられるファイルエントリの上記二次ポイントを、取って代わるファイルエントリを指すようにセットする手段とを具備し、この取って代わるファイルエントリは、取って代えられるファイルエントリの更新されたデータを含むことを特徴とするコンピュータファイル記憶システム。

【請求項31】 上記取って代わるファイルエントリの一次ポイントを上記取って代えられるファイルエントリの一次ポイントに等しくセットするための手段と、

上記取って代わるディレクトリエントリの兄弟ポイントを上記取って代えられるディレクトリエントリの兄弟ポイントに等しくセットするための手段とを更に備えた請求項30に記載のコンピュータファイル記憶システム。

【請求項32】 上記ファイル情報エントリは、最初に所定値にセットされる一次ポイント及び二次ポイントを含み、

ファイル情報エントリを記憶するためのメモリの一部分を割り当てる手段を更に備え、ファイル情報エントリは、ファイルのファイルエクステンツに関する情報を含むものであり、

更に、ファイルエントリの一次ポイントをその割り当てられたファイル情報エントリを指すようにセットする手段と、

ファイル情報エントリの上記一次ポイントを、その同じファイルに関連した別のファイル情報エントリを指すようにセットして、その同じファイルに対するファイル情報エントリのリンクされたりストを形成するための手段と、

取って代えられるファイル情報エントリの上記二次ポイントを、取って代わるファイル情報エントリを指すようにセットする手段とを具備し、この取って代わるファイル情報エントリは、取って代えられるファイル情報エントリの更新されたデータを含む請求項30に記載のコンピュータファイル記憶システム。

【請求項33】 上記取って代わるファイル情報エントリの一次ポイントを上記取って代えられるファイル情報エントリの一次ポイントに等しくセットする手段を更に備えた請求項32に記載のコンピュータファイル記憶システム。

【請求項34】 コンピュータメモリデバイスに記憶されたデータを新たなデータで更新する方法であって、上記メモリデバイスはデータの記録を含んでおり、各記録は最初に所定値を有する二次ポイントを備えており、上記方法は、

更新されるべきデータを含む記録を探索し、

新たなデータを含むように記録を割り当て、

その割り当てられた記録に新たなデータを書き込み、そして上記探索された記録における二次ポイントを上記割り当てられた記録を指すようにセットし、その割り当て

(5)

特開平7-191892

7

られた記録の新たなデータが上記探索された記録におけるデータの更新であることを指示するようにする、という段階を備えたことを特徴とする方法。

【請求項35】 二次ポインタをセットする上記段階は、二次ポインタが所定値から変化したことを示すように上記探索された記録にフラグをセットすることを含む請求項34に記載の方法。

【請求項36】 1つ以上の論理ブロックを有する一回書き込み多数回読み取りのメモリデバイスを備えたコンピュータシステム内でエラーを回復するための方法において、

(a) 各ブロックごとにデータ領域を画成し、データ領域は、データを記憶するためのデータ領域エリアに論理的に分割され、

(b) 各ブロックごとにブロック構造体を画成し、ブロック構造体はヘッダと割り当てテーブルとを有し、ヘッダはブロックに特定の情報を含み、割り当てテーブルは、データ領域エリアに対応するエントリを有すると共に、対応するデータ領域エリアに関連した情報を含んでおり、

(c) 割り当てテーブルエントリを割り当て、

(d) データを記憶するためのデータ領域エリアを割り当て、

(e) その割り当てられたデータ領域エリアに関連したデータをその割り当てられた割り当てテーブルエントリに書き込み、

(f) その割り当てられたデータ領域エリアにデータを書き込み、

(g) データ領域エリアに書き込みをしながらエラーを検出し、そして、

(h) エラーを検出した際には、割り当てテーブルエントリを割り当て解除状態にセットし、そしてデータ領域エリアにデータを書き込む間にエラーが検出されなくなるまで上記段階(c)、(d)、(e)及び(f)を繰り返すことを特徴とする方法。

【請求項37】 1つ以上の論理ブロックを有する一回書き込み多数回読み取りのメモリデバイスを備えたコンピュータシステム内でエラーを回復するための方法において、

(a) 各ブロックごとにデータ領域を画成し、データ領域は、データを記憶するためのデータ領域エリアに論理的に分割され、

(b) 各ブロックごとにブロック構造体を画成し、ブロック構造体はヘッダと割り当てテーブルとを有し、ヘッダはブロックに特定の情報を含み、割り当てテーブルは、データ領域エリアに対応するエントリを有すると共に、対応するデータ領域エリアに関連した情報を含んでおり、

(c) 割り当てテーブルエントリを割り当て、

(d) データを記憶するためのデータ領域エリアを割り

8

当て、

(e) その割り当てられたデータ領域エリアに関連したデータをその割り当てられた割り当てテーブルエントリに書き込み、

(f) その割り当てられたデータ領域エリアにデータを書き込み、

(g) その割り当てられた割り当てテーブルエントリを割り当て状態にセットし、

(h) その割り当てられた割り当てテーブルエントリを割り当て状態にセットしながらエラーを検出し、そして

(i) エラーを検出した際には、割り当てテーブルエントリをゼロ状態にセットし、そして割り当てられた割り当てテーブルエントリにデータを書き込む間にエラーが検出されなくなるまで上記段階(c)、(e)及び(g)を繰り返すことを特徴とする方法。

【請求項38】 消去可能な1つ以上の論理ブロックを有する一回書き込み多数回読み取りのメモリデバイスを備えたコンピュータシステム内でエラーを回復するための方法において、

上記ブロックに対してブロックの状態を含むブロックヘッダデータ構造体を画成し、

そのブロックヘッダデータ構造体にデータを書き込み、そしてそのブロックヘッダデータ構造体にデータを書き込む間にエラーが検出された際に、そのブロックを消去状態のための待ち行列にセットし、これにより、そのブロックは、それが消去されるまでデータの記憶に使用されないようにすることを特徴とする方法。

【請求項39】 消去可能な1つ以上の論理ブロックを有する一回書き込み多数回読み取りのメモリデバイスを備えたコンピュータシステム内でエラーを回復するための方法において、

上記ブロックに対してブロックの状態を含むブロックヘッダデータ構造体を画成し、

上記ブロックを消去し、そしてそのブロックの消去中にエラーが検出された際に、そのブロックをリタイア状態にセットし、これにより、そのブロックは、データの記憶に使用されないようにすることを特徴とする方法。

【請求項40】 コンピュータメモリマネージャにおいて指定のメモリデバイスがブロック消去可能でないことを指定する方法であって、上記コンピュータメモリマネージャはブロック消去可能な一回書き込み多数回読み取りのメモリデバイスをマネージするものであり、上記メモリマネージャはスベアブロックのカウントを維持し、更に、上記メモリマネージャは、データをスベアブロックにコピーし、指定されたブロックを消去しそしてその指定されたブロックをスベアブロックとなるようセットすることによってその指定されたブロック内のデータを圧縮するものであり、上記方法は、その指定のメモリブロックにおいてスベアブロックが使用できないために上記メモリマネージャがブロック内のデータを圧縮しない

(6)

特開平7-191892

9

ようにスベアブロックのカウントを0にセットする段階を備えたことを特徴とする方法。

【請求項41】 コンピュータメモリマネージャにおいて指定の一回書き込み多数回読み取りのメモリデバイスがブロック消去可能でないことを指定する方法であって、上記コンピュータメモリマネージャはブロック消去可能な一回書き込み多数回読み取りのメモリデバイスをマネージするものであり、更に、上記メモリマネージャは、ブロック内のデータを圧縮するのに使用するためのスベアブロックのカウントを維持し、上記方法は、メモリマネージャによりブロックの消去を抑制するようにスベアブロックの上記カウントを0にセットする段階を備えたことを特徴とする方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は、一般に、ファイルをマネージするためのコンピュータシステムに係り、より詳細には、フラッシュ消去可能なプログラマブル・リードオンリメモリ (FEPROM) に記憶されたファイルをマネージするための方法及びシステムに係る。

【0002】

【従来の技術】 コンピュータシステムは、一般に、揮発性及び不揮発性の両方の記憶装置における情報の記憶をサポートする。揮発性記憶装置と不揮発性記憶装置の相違は、揮発性記憶装置から電源が切断されたときに情報が失われることである。これに対して、不揮発性記憶装置から電源が切断されたときには、情報が失われない。従って、不揮発性記憶装置に情報を記憶した場合には、たとえコンピュータシステムの電源が切られていても、ユーザはあるときに情報を入力しそしてその後情報を探索することができる。又、ユーザは、不揮発性記憶装置をコンピュータから切断しそしてその記憶装置を別のコンピュータに接続して、その別のコンピュータが情報をアクセスするようにすることができる。

【0003】 不揮発性記憶装置に記憶された情報は、一般に、ファイルに編成される。ファイルとは、関連情報を収集したものである。時間の経過と共に、コンピュータシステムは、記憶装置の容量にもよるが、記憶装置に何百、何千のファイルを記憶することができる。情報を記憶することに加えて、コンピュータシステムは、典型的に、ファイルの情報を読み取ったり、修正したり、削除したりすることができる。コンピュータシステムは、記憶、読み取り、修正及び削除を効率的に行えるように記憶装置にファイルを編成することが重要である。

【0004】 一般にコンピュータのオペレーティングシステムの一部であるファイルシステムは、記憶装置上のファイルを管理する助けをするために開発されたものである。1つのこのようなファイルシステムが、マイクロソフト・コーポレーションによりそのディスクオペレーティングシステム (MS-DOS) として開発されて

10

いる。このファイルシステムは、ファイルを記憶するためにハイラーキ解決策を使用している。図1は、記憶装置のディレクトリ構造体を示している。これらのディレクトリはファイルの論理グループを含む。これらのディレクトリは、引出しの折り畳み器が引き出し内にペーパを編成していくのと同様にファイルを編成する。DOS、WORD、DAVID及びMARYと示されたブロックは、ディレクトリを表しており、そしてAUTOEXEC、BAT、COMMAND、COM、FORMAT、EXE、LETTER2、DOC、LETTER、DOCと示されたブロック及びLETTER1、DOCと示された2つのファイルは、ファイルを表している。このディレクトリ構造では、ユーザが関連ファイルをそれら自身のディレクトリに入れることによりファイルを編成することができる。この例において、WORDというディレクトリは、ワードプロセッシングプログラムWORDによって発生された全てのファイルを含む。このディレクトリWORDの中には2つのサブディレクトリDAVID及びMARYがあり、これらは、WORDファイルを、Davidにより開発されたもの及びMaryにより開発されたものへと更に編成する上で助けとなるものである。

【0005】 従来のファイルシステムは、不揮発性記憶装置の多数回書き込み機能の利点を取り入れている。多数回書き込み機能は、記憶装置上のいかなる情報ビットでも実質上無限の回数で1から0へそして0から1へ変更することができる。この機能では、ファイルを記憶装置に書き込みそしてファイルの幾つかのビットを変更することによりファイルを選択的に修正することができる。

【0006】 ディスクのような多数回書き込み能力を有する従来の記憶装置の欠点は、その速度が内部のコンピュータメモリの速度に比して遅いことである。これに対し、そのコンピュータメモリに勝るこれら記憶装置の利点は、不揮発性であること、コストが低いこと、容量が大きいことである。

【0007】 フラッシュEPROM (FEPROM) として知られている記憶装置は、内部コンピュータメモリの速度と、コンピュータディスクの不揮発性とは結合されたものである。この装置はEPROM型 (消去可能なプログラマブル・リードオンリメモリ) のデバイスである。FEPROMの内容は、典型的なEPROMと同様にデバイスに紫外線を照射するのではなく、入力にある電圧を印加することによって消去することができる。消去は、デバイス内の各ビットを同じ値にセットする。他のEPROMと同様に、FEPROMは不揮発性メモリである。FEPROMは、その速度がコンピュータの内部メモリに匹敵する。最初と、消去後とに、FEPROMの各ビットは1にセットされる。FEPROMの特徴は、他のEPROMと同様に、ビット値1を0に変更で

(7)

特開平7-191892

11

きるが、ビット値0は1に変更できないというものである。従って、データは、1から0へのビットの変更を行うようにE P r o mに書き込むことができる。しかしながら、いったんビットが0に変更されると、1に変更し直すことはできず、即ち全F E P r o mを全て1に消去しない限り1に戻すことができない。実際には、F E P r o mの各ビットは一度しか書き込むことができないが、次々の消去の間に何回も読み取ることができる。更に、F E P r o mの各ビットを消去して0にセットできるのは、ある限定された回数だけである。この限定された回数がF E P r o mの実効寿命を定める。

【0008】F E P r o mをアクセスする典型的な時間は、アクセスの形式と多数の他のファクタとによって異なる。読み取りアクセス時間は数百ナノ秒の範囲であり、バイトを読み取る回数については制限がない。書き込みアクセス時間は、典型的に数十マイクロ秒の範囲である。書き込みアクセス時間は、バイトを消去した回数と、デバイスの温度と、F E P r o mのバイト密度とによって左右される。バイトを書き込む回数に理論的な制限はないが、消去の制限が実際上の書き込みの制限となる。F E P r o mの消去時間は数秒の範囲である。消去時間は、F E P r o mを消去した回数、デバイスの温度、及びF E P r o mのバイト密度とによって左右される。

【0009】

【発明が解決しようとする課題】従来のファイルシステムは、記憶装置が多数回書き込み能力を有するものと仮定していたので、これらのファイルシステムは、実際上一回の書き込み能力しか持たないF E P r o mには不適である。従って、F E P r o mをベースとする記憶装置をサポートするファイルシステムをもつことが望ましい。このようなファイルシステムは、コンピュータシステムの速度と、コンピュータディスクの不揮発性とを有するものである。

【0010】コンピュータディスクのような従来の記憶装置は、バイトでアドレスできるのではなくブロックでアドレスすることができる。バイトは、コンピュータの内部メモリのアドレス能力の単位であり、即ちコンピュータは一度に1バイト（典型的に8ビット）で書き込み又は読み取りすることができるのであって、それ未満ではできない。コンピュータがディスクに書き込んだりディスクから読み取ったりするときには、ブロックと称するバイトのグループで行わねばならない。ブロックのサイズは変えられるが、典型的には2の累乗である（128、256、512等）。例えば、ディスク上の1バイトだけを変更しようとする場合に、そのブロックサイズ内の全バイト数の書き込みを行わねばならない。これには、ディスクからコンピュータメモリへブロック全体を読み取り、1バイトを変更し（内部メモリはバイトでアドレス可能である）そしてそのブロック全体をディスク

12

に書き込むことが含まれる。

【0011】従来のファイルシステムは、ブロックに未使用の部分を残す状態でデータを記憶する。このファイルシステムは、一度に1つのファイルのみからのデータを所与のブロック内に記憶する。例えば、このファイルシステムは、1つのファイルからのデータをブロックの最初の50バイトには記憶せず、そして別のファイルからのデータを128バイトブロックの最後の78バイトには記憶しない。しかしながら、ファイルの長さがブロックサイズの偶数倍でない場合には、ブロック端のスペースが未使用となる。上記例において、ブロックの最後の78バイトは未使用となる。ディスクが4096といった大きなブロックサイズを使用するときには、4095バイトまでのデータが使用できないことになる。多数回書き込み能力を有して数百万バイトを記憶できるようなディスクドライブではこのような未使用スペースが無視できる量であるが、多数回書き込み能力をもたず、しかも数百万バイトのデータを記憶する容量をもたない記憶装置では相当の量である。

【0012】F E P r o mは、典型的な記憶装置に比して、ブロックアドレス式ではなくてバイトアドレス式である。従って、F E P r o mのバイトアドレス能力をサポートするファイルシステムをもつことが望まれる。

【0013】又、F E P r o mは、ブロックで消去可能なフォーマットで編成することができる。ブロックで消去可能なF E P r o mは、個々に消去できる多数のブロック（典型的に16）を含んでいる。例えば、図6は、0から15まで番号が付けられた16個のブロックを有するブロック消去可能なF E P r o m 301の概略図である。ブロックの各々は、他のブロックの内容に影響を及ぼすことなく個々に消去することができる。ブロック番号14のブロック302は、他のブロックのデータに影響を及ぼすことなく消去することができる。ブロック消去可能なF E P r o mをサポートするファイルシステムをもつことが所望される。

【0014】そこで、本発明の目的は、ファイル記憶装置、特に、ブロック消去可能で且つフラッシュ消去可能なプログラマブル・リードオンリメモリにデータを記憶する方法を提供することである。

【0015】本発明の別の目的は、ブロック消去可能なF E P r o mにおいてメモリを割り当てたり割り当て解除したりするコンピュータメモリマネージャを提供することである。

【0016】本発明の別の目的は、ブロック消去可能なF E P r o mにおいてブロックを消去した回数を追跡する方法を提供することである。

【0017】本発明の更に別の目的は、メモリの割り当てを容易にするデータ構造体を備えたブロック消去可能なF E P r o mを提供することである。

【0018】本発明の更に別の目的は、データを記憶す



(8)

特開平7-191892

13

るためのブロックを割り当てる方法を提供することである。

【0019】本発明の更に別の目的は、ブロック消去可能なFEPromにおいて割り当て解除されたスペースをリクレーミングする方法を提供することである。

【0020】本発明の更に別の目的は、ブロック消去可能なFEPromのためのファイルシステムを提供することである。

【0021】

【課題を解決するための手段】本発明の上記及び他の目的は、以下の説明から明らかとなるように、ブロック消去可能で且つフラッシュ消去可能なプログラマブル・リードオンリメモリにおいてメモリをマネージするための方法及びシステムを提供することにより達成される。このシステムは、好ましい実施例において、ブロックヘッダと、ブロック割り当てテーブルと、データ記憶エリアと、データを記憶すべきブロックを選択するためのブロック割り当てルーチンと、ブロック割り当てテーブルのエントリ及びデータ記憶エリアの一部分を選択するためのデータエリア割り当てルーチンと、データを記憶するための記憶ルーチンとを有したブロック消去可能なFEPromを具備している。又、このシステムは、好ましい実施例において、ブロック消去可能なFEPromのためのファイルシステムを実施するファイルマネージャを備えている。

【0022】好ましい実施例において、本発明は、ブロック消去可能なFEPromのメモリをマネージする方法及びシステムを提供する。このシステムは、FEPromマネージャ及びファイルシステムとして説明する。FEPromマネージャは、FEPromのメモリの割り当てと割り当て解除とを管理する。ファイルシステムは、ハイアラキ式のディレクトリシステムであり、FEPromマネージャを用いてメモリの割り当て及び割り当て解除を行う。或いは又、FEPromマネージャ及びファイルシステムは、ある最適化を達成するように

14

一体化することができる。しかしながら、個別のFEPromマネージャを使用することによりFEPromは異なるファイルシステムからのデータや非ファイルシステムデータも記憶することができる。

【0023】

【実施例】本発明のFEPromマネージャは、ブロック消去可能なFEPromにおいて自由スペースの割り当て、割り当てられたスペースの割り当て解除、及び割り当て解除されたスペースのリクレーミングを行う。図26に示す好ましい実施例では、FEPromの各ブロックが、ブロック割り当て構造体2302と、データ領域2303と、自由スペース2304とを含んでいる。ブロック割り当て構造体2302は、ヘッダ及び割り当てアレイを含んでいる。ヘッダは、ブロックの状態情報を含む固定長さの構造体である。割り当てアレイは可変長さであり、割り当てアレイのエントリはデータ領域を記述する。テーブルAは、ブロック割り当て構造体のデータ構造を示している。この構造体は、構造体変数の記述と共にCプログラミング言語フォーマットで示されている。アレイのAllocは、割り当てアレイであり、他の変数がヘッダを構成する。データ領域は、FEPromに記憶されたデータを保持する可変長さの領域である。自由スペース2304は、ブロック割り当て構造体又はデータ領域に割り当てられないスペースである。ブロック割り当て構造体2302及びデータ領域2303はブロックの両端に割り当てられる。領域が追加されるときには、ブロック割り当て構造体2302及びデータ領域が矢印2305及び2306によって示されたように互いに向かって成長する。ブロック割り当て構造体のAllocエントリは、ブロックの対応領域に対するオフセット2310-2315を含んでいる。好ましい実施例においては、ブロック割り当て構造体が、その領域に記憶されたデータに対して特定のデータを含んでいる。

【0024】

テーブルA

データ構造体

struct BlockAllocation

```
{
    struct
    {
        byte    Status;
        byte    Offset[3];
        word    Len;
    }          Alloc [ ];
    dword     BootRecordPtr
    dword     EraseCount;
    word       BlockSeq;
    word       BlockSeqChecksum;
    word       Status;
```

(9)

特開平7-191892

15

16

}

定義

Alloc

ブロック内の領域を定める可変長さアレイ構造体

Status

領域の状態

ビット#

5-2

1111 未使用

1011 中間状態

0111 自由

0011 割り当てられた

0001 割り当て解除された

0010 取って代わられた

0000 ゼロ

7-6

11 未使用エントリ

10 最終エントリ

00 非最終エントリ

Offset

この領域のブロックの開始に対するオフセット

Len

領域の長さ(単位バイト)

BootRecordPtr

FEPromがファイル記憶装置として使用されるとき  
に記録をブートするための処理

EraseCount

ブロックを消去した回数

Blockseq

FEProm内のブロックの論理シーケンス

BlockSeqChecksum

ブロックシーケンス番号のチェック和

Status

ビット#

1-0

11 ブロックにないブート記録(FEProm がファイルシス  
テムデータを含むとき)

10 ブロックのブート記録

00 取って代わられたブート記録

15-10

110000 レディ

0XXXXX QueuedForErase

111111 消去された

111110 UpdateInProgress

111100 スベアブロック

111000 ReclamationInProgress

000000 リアタイアされた

【0025】FEPromマネージャは、Allocエントリを追加し、自由スペース内の第1位置を指すように可変オフセットをセットし、変数Lenを領域の長さにセットし、そして割り当てられたことを指示するように変数Statusをセットすることによりブロック内のデータ領域を割り当てる。FEPromマネージャは、対応するAllocエントリにおける変数Statusを割り当て解除された状態にセットすることにより領域の割り当て解除を行う。割り当て解除されたスペースは一般に再割り当てには使用できない。というのは、非FNULL値にセットされているからである。割り当て解除されたスペースは、再割り当てされる前にFNULLにセットされる。割り当て解除されたスペースをFNULLにセットしそして割り当てに使用できるようにするプロセスは、ブロックリクレーミングである。割り

当て解除されたスペースは、割り当てられた領域を第2のブロックにコピーしそして第2のブロックのAllocエントリを新たなオフセットを指すようにセットすることによりリクレーミングされる。ブロックのリクレーミングプロセスは、第2ブロックのAllocエントリがブロックヘッダに対しコピー元ブロックにあったのと同じ位置になるように確保する。FEPromマネージャは、論理ブロックシーケンス番号であるヘッダの変数BlockSeqを使用して、データの特定の論理ブロックを識別する。リクレーミング中に、ブロックがコピーされるときには、物理的なブロック番号が変化するが、論理的なブロックシーケンス番号は変化しない。

【0026】FEPromマネージャは、領域のスタートアドレスではなくて割り当てられた領域に対するハンドルを与える。このハンドルは、2つの部分、即ち

(10)

特開平7-191892

17

(1) 物理的なブロックを間接的に参照する論理ブロックシーケンス番号と、(2) 物理的なブロック内の領域を間接的に参照するA110cエントリに対するインデックスとを含んでいる。ハンドルは、論理的なブロックシーケンス番号に対応する物理的なブロックを決定しそしてハンドルのインデックス部分によって指示されたA110cエントリの変数Offsetをアクセスしてその変数Offsetを物理的なブロックのスタートアドレスに加えることにより参照解除される。この参照解除により領域のアドレスが形成される。ハンドルを使用すると、FEPromマネージャは、存在するかもしれないデータ領域へのリンクを調整せずにブロックをリクレーミングすることができる。更新しなければならないものは、メモリ内にあって論理ブロックシーケンス番号を物理的なブロックに対してマップする変換キャッシュ

(以下で説明する)と、A110cアレイのオフセットだけである。ハンドルが参照解除されるときには、新たなオフセットを用いて正しいアドレスが形成される。図29は、9の論理ブロックシーケンス番号2502と、3のA110cインデックス2503とを有するハンドル2501の参照解除を示している。ブロック変換キャッシュ2504は、論理的なブロック番号を物理的なブロック番号に対してマップする。FEPromマネージャはキャッシュを維持する。ある物理的なブロックがブロックのリクレーミング中に別の物理的なブロックへ移動されるときには、FEPromマネージャは、論理ブロックシーケンス番号を新たな物理的なブロックにマップするように変換キャッシュを調整する。図29の例においては、論理ブロックシーケンス番号9が物理的なブロック14 2505へとマップされる。物理的なブロック番号14に対するA110cアレイは、ハンドル2501のA110cインデックス2503によって指示される。A110c [3] エントリの変数Offsetは、物理的なブロック番号14 2505に領域3 2506のオフセットを含んでいる。領域3 2506のアドレスは、物理的なブロック2505のアドレスにオフセットを追加することによって決定される。

【0027】図33は、ブロック内に新たな領域を割り当てる領域割り当てルーチンの流れ線図である。このルーチンに対する入力パラメータは、その領域に記憶すべきデータと、データの長さである。このルーチンは、割り当てられた領域にハンドルを返送する。或いは又、このルーチンはデータを書き込まず、単にスペースを割り当てる。このルーチンは、ブロックが領域に対して十分な自由スペースと、追加のA110cエントリとを有すると仮定する。図35は、A110cエントリのステータスに対する状態図である。A110cエントリは、次のステータスのうちの1つである。即ち、未使用、割り当てられた状態、割り当て解除された状態、取って代わられた状態、自由、又はゼロである。又、エントリは、

18

割り当てが処理中である間は移行状態にある。エントリが未使用である場合には、A110cアレイの最後のエントリを通過しており、即ちアレイの一部ではない。エントリが割り当てられた場合には、それに対応する領域が現在割り当てられる。エントリが割り当て解除される場合には、それに対応する領域が不要であるが、その領域はまだリクレーミングされていない。エントリが取って代わられる場合には、別のA110cエントリ(単数又は複数)がその同じデータ領域に対応する。リクレーミング中、取って代わられたエントリにあるデータは無視される。というのは、別のエントリ(単数又は複数)がデータ領域を指すからである。エントリが自由の場合には、それに対応する領域がリクレーミングされており、新たな領域がブロックに追加されたときにA110cエントリを再使用することが出来る。エントリがゼロの場合には、エントリへの書き込み中に問題が生じており、消去されるまで使用されない。エントリが割り当てにおいてプロセス移行状態にある場合には、エントリにおけるデータの若干は有効であるが必ずしも全部ではない。

【0028】図35を参照すれば、全てのエントリは最初は未使用状態3101にあり、ブロックが消去されたときに未使用状態3101へ移行する。エントリは、未使用状態3101から割り当て進行状態3104を経て割り当て状態3103へ移行する。自由状態3102にあるエントリは割り当て状態3103へ移行する。未使用状態3101にあるエントリは、そのエントリが割り当て解除されるか又は取って代わられて、リクレーミングされるべきブロック内の最後に割り当てられたエントリの後に探索されないときに、ブロックリクレーミングによって自由状態3102へ移行する。割り当てられた状態3103にあるエントリは、対応する領域が割り当て解除されたときに割り当て解除状態3105に移行する。割り当てられた状態3105にあるエントリは、同じ領域に対応する別の1つ又は複数のエントリが割り当てられたときに、取って代わられた状態3106へ移行する。最後に、いかなる状態にあるエントリも、そのエントリに対する書き込みエラーの際にはゼロ状態3107へ移行する。

【0029】図33を参照すれば、ブロック2901において、システムはA110cエントリが自由の状態にあるかどうかを判断する。このようなエントリが見つかった場合には、システムはそのエントリを再使用し、ブロック2902へ続くが、さもなければ、ブロック2903へ続く。ブロック2902において、システムは自由のA110cエントリを選択し、ブロック2905へ続く。ブロック2903において、システムは新たなA110cエントリを選択し、ブロック2904へ続く。新たなA110cエントリは、最終とマークされたエントリの直後のエントリである。ブロック2904におい

(11)

特開平7-191892

19

て、システムは、選択されたA l l o c エントリに対して変数S t a t u s をセットし、割り当てが進行中であることを指示する。割り当て進行中状態は、データが一貫した状態にないかもしれないことを指示する移行状態である。ブロック2905において、システムは変数O f f s e t 及びL e n をセットし、自由スペースの第1位置から始まってデータ領域にデータを書き込む。ブロック2906において、A l l o c エントリが新規であった場合には、システムはブロック2907へ続くが、さもなくば、システムはブロック2908へ続く。ブロック2907において、システムは手前の最終A l l o c エントリのステータスをリセットし、もはやA l l o c 構造体の最終エントリでないことを指示する。ブロック2908において、システムは、選択されたA l l o c エントリに対する変数S t a t u s を割り当てられた状態にセットし、データが今や一貫した状態にあることを指示する。次いで、システムは割り当てを終了する。

【0030】上記したように、F E P r o m のブロックの性能は、消去カウントが増加するにつれて低下する。各ブロックの消去カウントをほぼ等しく（均等化したと称する）維持することが好ましい。通常の動作においては、消去カウントが均等化されない。例えば、実行可能なファイルがブロックに書き込まれた場合には、そのブロックを消去する必要は決して生じない。従って、そのブロックの消去カウントは、他のブロックの消去カウントが増加しても一定のままである。好ましい実施例ではブロックの消去カウントを均等化するために多数の解決策を使用する。まず、ブートアップ中又はF E P r o m がロードされるときに（初期化）、F E P r o m マネージャはブロックを走査して、F E P r o m をマネージするデータを得る。好ましい実施例では、このデータが各ブロックの消去カウントを含んでいる。消去カウントの均等化を助けるために、F E P r o m マネージャは、消去カウントの大きいブロックのデータと、消去カウントの低いブロックのデータをスワッピングする。このスワッピングは時間がかかるので、初期化中に生じるスワッピングの回数を最小にすることが望ましい。更に、スワッピングは、消去カウントの差がスレッシュホールド値又は比を越えたときだけ実行すればよい。第2に、F E P r o m マネージャがブロックに対してリクレーミングを実行するときには、消去カウントの小さい使用可能なブロックを選択するのが好ましい。第3に、F E P r o m マネージャが領域を割り当てるときには、消去カウントの低いブロックの領域を割り当てる。第4に、F E P r o m マネージャはブロックの消去回数を追跡する。消去の回数がスレッシュホールド値を越えた場合には、F E P r o m マネージャは、2つのブロックに対する消去カウントの差がスレッシュホールド値又は比を越えたかどうかを決定する。そのスレッシュホールド値を越えた場合には、マネージャはそれらブロックのデータをスワ

20

ッピングする。

【0031】F E P r o m マネージャは、各物理的なブロックのヘッダに消去カウントを維持するのが好ましい。ブロックが消去されたときには、F E P r o m マネージャは、増加された消去カウントをブロックヘッダに書き込んで戻す。ブロックがコピーされたときには、消去カウントが転送されない。各ブロックはそれ自身の消去カウントを保持する。或いは又、消去カウントは単一のブロックに記憶することができる。しかしながら、この別の方法は、好ましい方法以上に幾つかの欠点がある。第1に、単一のブロックに欠陥が生じると、全ての消去カウントが失われる。第2に、ブロックが消去されると、消去カウントブロックを更新しなければならない。最終的に、消去カウントブロックは消去され、再書き込みされねばならない。

【0032】図32は、ブロック消去可能なF E P r o m のどのブロックを使用して領域を割り当てるかを選択するブロック割り当てルーチンの流れ線図である。F E P r o m マネージャは、多数のファクタに基づいてどのブロックを割り当てるかを決定する。まず、F E P r o m マネージャは、消去カウントが最も低い充分な自由スペースを有するブロックを割り当てる。消去カウントの低いブロックを割り当てることは、ブロックの均等化を確保する上で助けとなる。第2に、F E P r o m マネージャは、充分な自由スペースをもつブロックがない場合には多数のブロックを割り当てる。データは、異なるブロックに各々記憶された多数の領域に分割される。第3に、F E P r o m マネージャは、あまりに多数の部分が生じるときには割り当てを許さない。このルーチンに対する入力パラメータは、割り当てられるべき領域の長さ、領域を多数のブロックに記憶できるかどうかである。ブロック2801において、システムは、データを記憶するに充分な自由スペースを有する全てのブロックを選択する。好ましい実施例において、システムは、自由スペースの開始位置とA l l o c エントリの数とに基づいて自由スペースの長さを決定する。このデータは、初期化中及び必要に応じて更新される間にF E P r o m マネージャバッファに記憶されるのが好ましい。又、システムは、もし必要ならば、新たなA l l o c エントリを追加するに充分なスペースが存在するよう確保する。1つの実施例において、システムは、ブロックが充分な自由スペースを有しているかどうかを決定する前に、リクレーミング基準に合致するブロックに対してブロックリクレーミングを実行する。別の実施例においては、充分な自由スペースがないと決定されるまでリクレーミングは行われない。ブロック2802においては、少なくとも1つのブロックが選択された場合に、領域を保持するに充分な自由スペースが単一ブロックにあり、システムはブロック2803に続き、さもなくば、システムはブロック2804に続く。ブロック2803において、

(12)

特開平7-191892

21

システムは、選択されたブロックのどれが最も低い消去カウントを有するかを決定し、そのブロックを割り当てると共に、システムはブロックの割り当てを終了する。ブロック2804においては、システムは、領域データを保持するに十分な自由スペースを有する1組のブロックを選択する。ブロック2805においては、全自由スペースと割り当て解除されたスペースがデータを保持するに充分でないか或いはあまりに多数の部分がある場合に、システムはブロック2807へ続き、さもなければ、システムはブロック2806へ続く。領域データを単一ブロックに記憶しなければならない場合には、2つのブロックを選択すると、部分が多数になり過ぎる。又、データを多数のブロックに記憶すべき場合には、リクレーミングが適当である。ブロック2806において、システムは選択されたブロックを割り当て、ブロックの割り当てが終了する。ブロック2807において、全てのブロックがリクレーミングされた場合には、FEPROMにはデータを記憶するに十分な余裕がなく、ブロックの割り当てが終了するか、さもなければ、システムはブロック2808へ続く。ブロック2808において、システムはブロックをリクレーミングし、ブロック2801へ進んで、新たな充分な自由スペースがあるかどうかを決定する。

【0033】図34は、ブロック内の割り当て解除された領域をリクレーミングするブロックリクレーミングルーチンの流れ線図である。入力パラメータは、リクレーミングされるべきブロックの数と、スベアブロックの物理的な番号とである。ブロックは、多数の種々の時間にリクレーミングされる。第1に、割り当て要求を満たす充分な自由スペースがないときには、ブロックがリクレーミングされる（上記したように）。第2に、FEPROMマネージャは、FEPROMへの書き込みアクセスの回数を追跡することができる。書き込み回数がスレッシュホールド数を越えた場合には、マネージャがいずれかのブロックをリクレーミングすべきかどうかを決定する。割り当て解除されたスペースとブロックサイズとの比がスレッシュホールド値を越えたときにはブロックをリクレーミングしなければならない。当業者に明らかのように、例えば、FEPROMが最初にロードされるときのような別の時間にブロックリクレーミングを行うこともできる。FEPROMマネージャは、割り当てられた領域をスベアブロック、即ち消去されたブロックへコピーすることによりブロックをリクレーミングする。割り当てられた領域のみをコピーすることにより、割り当て解除された領域がリクレーミングされる。或いは又、FEPROMマネージャは、割り当てられた領域を非FEPROMメモリへコピーし、次いで、ブロックを消去し、そして領域をブロックへコピーして戻す。しかしながら、この方法は、割り当てられた領域を記憶するに充分な非FEPROMメモリを必要とし、消去の後であっ

22

て且つブロックが書き込みされる前に停電が生じたとすれば、データを失うおそれがある。好ましい方法においては、FEPROMマネージャは、リクレーミングされるべきブロック内の割り当てられた領域をスベアブロックへコピーすると共に、そのスベアブロック内の新たな領域位置を表す変数Offsetを調整するブロック割り当て構造体をコピーする。図27は、領域1及び5をリクレーミングした後の図26のブロックレイアウトの例を示している。割り当てられた領域は隣接するようにコピーされている。対応するAllocエントリは、新たな領域位置を指すように更新される。たとえば領域1がリクレーミングされていてもAlloc[1]エントリは依然として必要とされる。ハンドルを使用するために、全てのAllocエントリはブロック割り当て構造体においてそれらの同じ位置を維持しなければならない。しかしながら、Alloc[1]エントリに対する変数Statusは自由状態にセットされ、これを用いて、リクレーミングされたブロックに追加される次の位置を指すことができる。Alloc[5]エントリの後にはAlloc入力はないので、ハンドルに対するブレースホルダーとして必要とされず、除去されている。Alloc[4]エントリの位置状態は、それがAllocアレイにおける最終エントリであることを指示する。

【0034】図36はブロックの状態を示す状態図である。ブロックの状態は変数Statusのヘッダに記憶される。ブロックの状態は、消去3201、更新進行中3202、スベア3203、リクレーミング進行中3204、レディ3205、消去のための待ち行列3206、及びリタイア3207である。新たに消去されたブロックは消去状態3201である。消去されたブロックは、通常、更新進行状態3202へ移行され、次いで、スベア状態3203へ移行される。更新進行状態3202は、ヘッダ内のあるデータ、例えば消去カウントが更新されていることを指示する。この更新が完了すると、ブロックはスベアの状態3203へ移行する。更新が失敗すると、ブロックは消去のための待ち行列状態3206へ移行する。スベア状態3203のブロックは、そのブロックがリクレーミングされているブロックからデータを受け取るべきときに、リクレーミング進行状態3204へ移行する。リクレーミング進行状態3204は、ブロック割り当て構造体が一貫した状態にないことを指示する移行状態である。データが一貫したものになると、ブロックはレディ状態3205へ移行する。しかしながら、リクレーミング進行状態3205の間にエラーが生じた場合には、そのブロックに対してリクレーミングが行われた後に、ブロックが消去待ち行列状態3206へ移行する。消去待ち行列状態3206にあるブロックは、消去されたときに消去状態3201へ移行する。消去が失敗すると、ブロックはリタイア状態3207へ移行する。それからブロックはリタイア状態に留まる。

23

FEPromが最初に初期化されるときには、ブロックがレディ状態又はスベア状態にセットされる。レディ状態にあるブロックはデータを含むことができ、スベア状態にあるブロックはデータを含まない。

【0035】図34を参照すれば、ブロック3001において、システムは、リクレーミングされるべきスベアブロックに対する変数Statusを、リクレーミング進行中を指示するようにセットする。ブロック3002において、システムは、リクレーミングされるべきブロックのヘッダから変数Seq、SeqCheckSum、及びBootRecordPtrをスベアブロックヘコピーする。ブロック3003において、システムは、リクレーミングされるべきブロックに対して割り当て状態にある最終Allocエントリの位置を決定する。リクレーミングプロセス中に、Allocエントリは、最後に割り当てられたエントリの後に無視される。従って、リクレーミングされたブロックは、これらエントリの状態を未使用にセットする。ブロック3004ないし3010において、システムは、最後に割り当てられたエントリまでの各Allocエントリをスベアブロックにコピーする。ブロック3004において、システムは、Allocエントリを指示するインデックスjを初期化する。ブロック3005において、このインデックスjがブロック3003で決定された最後に割り当てられたエントリのインデックスより大きい場合には、全てのAllocエントリとそれに対応する領域がコピーされており、システムはブロック3011へ続き、さも

20

なくば、システムはブロック3006へ続く。ブロック3006において、エントリの状態が割り当てられた状態である場合には、システムはブロック3007へ続き、さも

30

なくばブロック3009へ続く。ブロック3007において、システムは、Alloc{j}エントリに対応する領域データをスベアブロックヘコピーする。ブロック3008において、システムは、スベアブロックのAlloc{j}入力における変数Offsetを更新し、コピーされた領域の位置を指示するようにすると共に、変数Status（位置の状態を適当にセットする）及びLenをコピーし、ブロック3010に続く。ブロック3009では、システムは、スベアブロックのAlloc{j}エントリ

40

の状態を更新し、そのエントリが自由状態であることを指示するようにする。ブロック3010では、システムはインデックスjを増加して、次のAllocエントリを指示し、ブロック3005へとループする。ブロック3011においては、システムはスベアブロックの状態をレディにセットする。ブロック3012では、システムは、リクレーミングされるべきブロックの状態を消去のための待ち行列にセットする。ブロック3011についての処理が完了した後であって且つブロック3012についての処理が完了する前に、スベアブロックと、リクレーミングされるべき

50

24

ブロックの両方は有効データを有している。ブロック3012についての処理が完了する前に処理が中断した場合には、FEPromは、同じ論理シーケンス番号を有する2つのブロックを含む。システムは、好ましくは、FEPromの初期化中にこの状態をチェックする。このときに、システムはブロック3012の処理を完了することができる。ブロック3013では、システムは、スベアブロック（以下で述べる）の状態を表すように変数PhysicalBlockNum、FirstFreeByteOffset、LenDeallocatedSpace及びAllocStructCnt in BlockData{Seq}を更新する。ブロック3014では、システムは、スベアブロックのリストを調整するようにDriveRecを更新し、リクレーミングが終了となる（以下で述べる）。

【0036】

表 B

データ構造	
struct DriveRec	
{	
word BlockCnt	
word SpareBlockCnt	
dword BlockSize	
word RootDirPtr	
word SpareBlockPtr [ ]	
}	
struct ConfigRec	
{	
word WriteAccessCntThreshold	
word EraseCntThreshold	
word BlockReclamationThreshold	
word BlockEraseLevelingThreshold	
}	
struct BlockRec	
{	
byte Flags	
word PhysicalBlockNum	
dword FirstFreeByteOffset	
dword LenDeallocatedSpace	
word AllocStructCnt	
dword BlockEraseCnt	
word FirstUseableAllocEntry	
word FreeAllocEntryCnt	
}	
BlockData [ ]	
定義	
BlockCnt	FEProm内の物理ブロックの数
BlockSize	ブロック内のバイトの数
RootDirPtr	FEPromをファイル記憶装置として使用した時のルートディレクトリを含むデー

(14)

特開平7-191892

25

タ領域へのハンドル

SpareBlockPtr [ ] 予備ブロックの物理ブロックの数を含む可変長アレイ

SpareBlockCnt 予備ブロックの数

WriteAccessCntThreshold ブロックを再利用すべきか否かをシステムに決定させる FEProm への書き込みの数

EraseCntThreshold ブロックレベル化を行うべきか否かをシステムに決定させる FEProm への消去の数

BlockReclamationThreshold ブロック再利用をトリガする割当て解除スペースとブロックサイズとの比

BlockBraceLevelingThreshold ブロック間のレベル化プロセスをトリガする最小及び最大消去数間の差

PhysicalBlockNum 論理的ブロックを含む物理ブロックの数

FirstFreeByteOffset 空きスペースの最初のバイトの物理ブロック内のオフセット

LenDeallocatedSpace 物理ブロック内の割当て解除済領域の合計の長さ

FirstUseableAllocEntry ブロック内の最初の使用可能な Alloc エントリ索引

FreeAllocEntryCnt Alloc エントリの数

BlockEraseCnt 物理ブロックが消去された回数

FEPromが最初にロードされる時、FEProm管理者は FEProm を走査して表Bに示す内部データを初期化する。構造 DriveRec は装置に関するデータを含み、構造ConfigRec は構成パラメタに関するデータを含み、アレイ BlockDataは FEProm内の各物理ブロック毎のデータを有するエントリを含む。アレイ BlockDataはブロック変換キャッシュである。初期化中、FEProm 管理者はアレイ BlockData内の各変数及び構造 DriveRec 内の予備ブロックに関する変数を初期化する。構造DriveRec内の他の変数はシステム定義された変数である。好ましい実施例では、FEProm管理者は、これらのデータ構造内の領域データの型に特定の情報を記憶している。例えばもし FEProm がファイル記憶装置として使用されていれば、データ構造はルートディレクトリへのハンドルを含むことができる。図31は、好ましい実施例における初期化プロセスの流れ図である。この手順は、FEProm 上の各ブロック毎にブロック割当て構造を走査することによって DriveRec 及び BlockData構造を初期化する。システムはブロック2701乃至2709をループして各ブロック毎のデータを読む。ブロック2701ではシステムは、現在アクセスされている物理ブロックを指示する索引1を初期化する。ブロック2702において、もし索引1が FEProm 内のブロックの数より大きければ、全てのブロックは走査されたのでありシステムはブロック2710へ進み、そうでない場合にはシステムはブロック270

26

3へ進む。ブロック2703では、システムは索引1によって指示されたブロックの見出しを読む。ブロック2704では、システムは DriveRec 及び BlockData [i] データを更新する。もしそのブロックが予備ブロックであれば、システムは SpareBlockCntをインクリメントさせ、そのブロックを SpareBlockPtrアレイに追加する。好ましい実施例では、システムはこれらの領域内に記憶されているデータに特定の情報をも走査する。例えばもし FEProm をファイルシステムとして使用している、そのブロックがブートレコードを含んでいれば、システムは BootRecPtr をセットし、ブートレコードを読み、そして RootDirPtr をセットする。

【0037】システムはブロック2705乃至2708をループして各 Alloc エントリ内のデータを処理する。ブロック2705では、Alloc エントリを指示する指標jを初期化する。ブロック2706において、もしシステムが最後のAlloc エントリを処理済であればシステムはブロック2709へ進み、そうでなければシステムはブロック2707へ進む。ブロック2707では、システムはjによって指示された Alloc エントリに基づいて BlockData [BlockSeq] データを更新する。システムは変数 FirstFreeByteOffset、LenDeallocatedSpace、及び AllocStructCntを更新する。システムは、変換キャッシュを初期化する索引iに変数 PhysicalBlockNum をセットする。ブロック2708では、システムは索引jをインクリメントさせて次の Alloc エントリを指示させ、ブロック2706へループする。ブロック2710では、システムはブロック使用をレベル化する。システムは BlockDataアレイを走査して最大 BlockEraseCntを有するブロック及び最小 BlockEraseCntを有するブロックを決定する。次いでシステムはブロック間でデータをスワップ（交換）する。システムは先ず最大ブロックを予備ブロックへ複写する。次にシステムは最大ブロックを消去する。システムは最小ブロックからのデータを消去されたブロックへ複写し、そして好ましくは複写しながらブロック再利用を遂行する。システムは最小ブロックを消去し、予備ブロックからのデータを最小ブロックへ複写し、そして好ましくは複写中にブロック再利用を遂行する。

【0038】本発明の FEProm 管理者はブロック化されていない、または消去することができない媒体を支援することができる。ブロック再利用及びブロック消去カウトレベル化プロセスはブロック消去可能性に頼っている。従ってもし媒体がブロック消去可能性を支援しなければ、これらのプロセスを不能にすべきである。好ましい実施例では、予備ブロックカウントを0にセットすることによってこれらのプロセスを実効的に不能にすることができる。FEProm 管理者はこれらのプロセスを作動させるために少なくとも1つの予備ブロックに頼っている。もし媒体がブロック化されていなければ、任意の

(15)

特開平7-191892

27

ブロックサイズを論理的ブロックとして選択することができる。好ましい実施例ではブロックのサイズは、割当てアレイエントリ内のオフセットが全ブロックをアドレスできない程大きくすべきではなく、またブロック見出し及び割当てアレイがブロックサイズのかなりなパーセントを占めたり、または変換キャッシュが大き過ぎる程は小さくすべきでない。

【0039】FEProm 管理者は、FEProm 書き込み及び消去エラーからの動的な回復を可能にする。書き込みエラーは、メモリ位置を指定された値にセットできない場合に生成される。これらのエラーは、ハードウェアの障害によって、または既に0に変化しているあるビット内に1を要求するというように、あるメモリ位置にある値を書き込もうと試みることによってもたらされる。

【0040】書き込みエラーはデータ領域、ブロック見出し、及び割当てアレイエントリへ書き込む時に発生し得る。好ましい実施例では、データ領域へ書き込み中に書き込みエラーが発生すると、FEProm 管理者はそのブロックを割当て解除された状態にセットする。次いで FEProm 管理者は、図33に示す領域割当てプロセスを再始動させることによって、データを異なるデータ領域へ書き込むことを試みる。

【0041】割当てアレイエントリへ書き込み中に書き込みエラーが発生すると、FEProm 管理者はその割当てアレイエントリを空白(ヌル)状態にセットする。あるエントリを置換された(superseded)状態、割当て解除された状態、解放された(フリー)状態、または割当て進行中状態にセットしている時に、もし書き込みエラーが発生すると、そのエントリを空白状態にセットすることによって FEProm は一貫した(consistent)状態に保たれる。しかしながら、もしあるエントリを割当て済状態にセットしている時にエラーが発生すると、データ領域は割当て済状態の対応割当てアレイエントリを有していないのであるから、FEProm は非一貫状態になる。FEProm 管理者はそのデータ領域に対して別のエントリを割当てる。あるエントリを空白状態にセットしている時にもエラーは発生し得る。空白状態は、0の値のステータスとして定義されているから、あるエントリを空白状態にセットしている時に発生するエラーは必然的にハードウェアエラーである。もしエラーが発生すれば、対応する領域を再利用しなければならぬことを指定する再利用を必要とするかも知れない。例えば、もしあるエントリを割当て解除状態にセット中に、及び再びそのエントリを空白状態にセット中にエラーが発生すれば、そのエントリは割当て済状態になる。もしこの状態のままであれば、このエントリ及び対応データ領域は正常の再利用の下では決して再利用されなくなる。

【0042】ブロック見出しに書き込み中にエラーが発生すると、FEProm 管理者はそのブロックを消去待ち行列状態にセットする。もしあるブロックを消去待ち行列状

28

態にセット中にエラーが発生すれば、FEProm 管理者はそのブロックを引退(retired)状態にセットするので、そのエラーを回復することはできなくなる。あるブロックを消去中に書き込みエラーが発生すると、FEProm 管理者はそのブロックを引退状態にセットする。もし引退させたブロックが予備ブロックであったならば、FEProm 管理者は少なくなった予備ブロックで動作する。代替としてFEProm管理者は割当て済の割当てアレイエントリを用いずにブロックを探知することを試みる。次いで FEProm 管理者は探知したブロックを消去し、それを予備状態にセットする。予備ブロックが使用できない場合には FEProm 管理者は FEProm を、あたかもそれが前述した消去不能であるかのように取扱う。

【0043】本発明は、FEProm が非一貫状態にある場合の動的エラー回復をも提供する。図33を参照する。例えば、もしブロック2905においてオフセットが書き込まれた後に、しかしブロック2908において状態が解放状態から割当て済状態へ更新される前に FEProm が除去されれば、FEProm は非一貫状態になる。FEProm に次にロードする時に、FEProm 管理者は割当てエントリが解放されていることを見て、それを再使用しようとする。しかしながら、オフセットへ書き込もうとする試みは失敗する(同一データがオフセットへ書き込まれている場合を除く)。上述したように、FEProm 管理者はエントリを空白状態にセットすることによって回復し、領域割当てプロセスを再始動させて異なるエントリを選択する。もしFEPromが取外される前にデータの一部分がデータ領域へ書き込まれれば、FEProm 管理者がデータをその領域へ書き込もうとする時にエラーが検出される。このエラーは上述したようにして処理される。

#### ファイルシステム

本発明は、FEProm 装置に対してディレクトリをベースとする階層ファイルシステムを提供する。階層ファイルシステムは論理的グルーピング内にファイルを記憶させる。好ましい実施例は、ディレクトリ階層及び内部ファイル記憶の両者を実現するためにリンクされたりリストデータ構造を使用する。

【0044】図1に典型的な階層ディレクトリ構造を示す。ワシントン州レッドモンドのマイクロソフト社から入手できる MS-DOS オペレーティングシステムが、階層ディレクトリ構造を有するファイルシステムを実現する。図1に示すように、ディレクトリ「ルート」100は、2つのサブディレクトリ(「DOS」102及び「ワード」103)と、2つのファイル(「AUTOEXEC.BAT」104及び「COMMAND.COM」105)を含む。ディレクトリ「DOS」102は1つのファイル(「FORMAT.EXE」106)を含む。ディレクトリ「ワード」103は次に低いレベルに2つのサブディレクトリ(「デービッド(DAVID)」107及び「メリー(MARY)」108)を含む。ディレクトリ「デービッド」107は1つ



(16)

特開平7-191892

29

のファイル「LETTER1.DOC」109を含む。ディレクトリ「メリー」108は3つのファイル「LETTER1.DOC」110、「LETTER2.DOC」111及び「LETTER3.DOC」112を含む。

【0045】図2は、好ましい実施例において図1のディレクトリ構造を実現する考え得るリンクされたリストを示す。ディレクトリ「ルート」レコード100（本明細書においては「レコード」及び「エントリ」という語を互換可能なように使用する）はポインタ120を有し、このポインタ120は次に低いレベルのサブディレクトリのリンクされたリスト140及びファイルレコードを指し示す。リンクされたリスト140は、ポインタ121、122、123によってリンクされたサブディレクトリレコードDOS102及び「ワード」103と、ファイルレコード「AUTOEXEC.BAT」104及び「COMMAND.COM」105とからなる。サブディレクトリレコード「DOS」102は次に低いレベルのファイルレコード106を指し示すポインタ124を有し、サブディレクトリレコード「ワード」103は次に低いレベルのファイルレコードのリンクされたリスト141を指し示すポインタ125を有している。リンクされたリスト141は、ポインタ126によってリンクされているディレクトリレコード「デービッド」107及び「メリー」108からなる。サブディレクトリレコード「デービッド」107は次に低いレベルのファイルレコードを指し示すポインタ127を有し、サブディレクトリレコード「メリー」108は次に低いレベルのファイルレコードのリンクされたリスト142を指し示すポインタ128を有している。リンクされたリスト142は、ポインタ129及び130によってリンクされているファイルレコード「LETTER1.DOC」110、「LETTER2.DOC」111及び「LETTER3.DOC」112からなる。右上のテンプレート10は図面を通して使用されるレコードレイアウトを示す。好ましい実施例では図2に示すレコードは、以下に説明するように DirEntry 及び FileEntry 構造である。

【0046】図2は、図1を表す単に1つの考え得るリンクされたリスト配列を表すに過ぎない。この配列は、もしファイルが追加されたが次いで削除されたか、またはディレクトリ名が変更されていれば異なる配列になる。図3は別の考え得る配列を示す。図3は、図1と同一のディレクトリ階層を表しているが、一時期存在したディレクトリ「ビル(BILL)」113は削除されている。FEProm 装置は一回取り書き込み可能である（消去された場合を除く）から、本発明の好ましい実施例では図3に示すように、ディレクトリレコード「ビル」113をリンクされたリストから物理的に除去しない。ディレクトリまたはファイルは、ディレクトリまたはファイルエントリのステータスをセットすることによってリンクされたリストから削除される。もしディレクトリまたはファイルがコンピュータディスク上に記憶されていたの

30

であれば、ディレクトリレコード「ビル」113は、ディレクトリレコード「メリー」108を指し示しているポインタ131をディレクトリレコード「デービッド」107内に再書き込みすることによって物理的に除去することができる。

【0047】好ましい実施例は、あるファイルを構成する範囲(extent)をリンクするためにもリンクされたリストデータ構造を使用する。各ファイルはそれに対応付けられたファイルレコードを有し、このファイルレコードは他のデータと共にファイル名を含み、また上述したようにディレクトリ階層内にリンクされている。ある範囲は、そのファイルに関するデータを含むメモリの連続域である。各ファイルは、ファイルデータを含む1またはそれ以上の範囲である。各範囲はそれに対応付けられた範囲レコードを有している。範囲レコードは、他のデータと共にその範囲を指し示すポインタと範囲の長さを含む。図4はファイル「\A\D.DAT」202の範囲を示す。範囲レコードR1 203、R2 204及びR3 205はリンクされ、対応範囲E1 211、E2 212及びE3 213を指し示すポインタを含んでいる。ファイルは範囲E1 211、E2 212及びE3 213の論理的連結である。好ましい実施例では、範囲レコードは後述するように FileInfo 構造である。

【0048】図4は、ファイル「\A\D.DAT」202のための単なる1つの考え得るリンクされたリスト配列を表すに過ぎない。図5に同一ファイルを表す別の配列を示す。範囲E4 214はファイルに追加されたが、削除されている。好ましい実施例では範囲レコードR4 206は、ファイルを構成する範囲のリンクされたリストから物理的に除去されていない。そうではなく、レコードの削除を指示するようにレコードのステータスをセットすることによって範囲レコードR4 206は論理的に除去されているのである。

【0049】表C及びDは、本発明の好ましい実施例に使用される幾つかのデータ構造を含んでいる。表Cに示されているデータ構造は BootRecord である。BootRecord は、以下に説明するように、ファイルシステムの識別に関するある一般的な情報、FEProm をアクセスすることができるファイルシステムのバージョン番号、ルートディレクトリを指し示すポインタ、及び付加的なデータを含む。表Dに示す第1及び第2の構造は、DirEntry 構造及び FileEntry 構造である。これらの構造の1つが各ディレクトリ及びファイル毎に割当てられる。これらの構造は同一である。変数 SiblingPtr はディレクトリ階層の同一レベルの DirEntry 構造及び FileEntry 構造のリンクされたリスト内の次の同胞を指し示す。変数 PrimaryPtr 及び SecondaryPtr は以下に詳述する。第3の構造は、FileInfo 構造である。各ファイル範囲は関連 FileInfo 構造を有している。変数 PrimaryPtr は

(17)

特開平7-191892

31

32

ファイルの FileInfo 構造を指し示す。

表 C

表 D

データ構造	
struct BootRecord	
{	
word	Signature;
dword	SerialNumber;
word	FPSWriteVersion;
word	FPSReadVersion;
word	TotalBlockCount;
word	SpareBlockCount;
dword	BlockLen;
dword	RootDirectoryPtr;
word	Status;
byte	VolumeLabelLen;
word	BootCodeLen;
byte	VolumeLabel [ ];
byte	BootCode [ ];
}	
定義	
Signature	媒体がこのファイルシステムを 20
支援することを指示する値	
SerialNumber	VolumeLabel との組合せは特定
FEProm の独特な識別子である	
FPSWriteVersion	このボリュームへ書き込むために
要求されるファイルシステムの高バイトにおけるバージ	
ョン番号及び低バイトにおける改定番号	
FPSReadVersion	このボリュームを讀出すために
要求されるファイルシステムの最早バージョンの高バイ	
トにおけるバージョン番号及び低バイトにおける改定番	
号	
TotalBlockCount	FEProm内の予備ブロックを含む
ブロックの合計数	
SpareBlockCount	ブロック再利用及びエラー回復
に使用可能なブロックの数	
BlockLen	バイトで表したブロックの長さ
RootDirectoryPtr	ルートディレクトリを指し示す
ポインタ	
Status	ファイル名フォーマットを指定
するデータ	
VolumeLabelLen	ボリュームラベル内の文字の数 40
BootCodeLen	ブートコードアレイ内のバイト
の数。もし0ならば媒体はブート不能	
VolumeLabel [ ]	ボリュームラベル
BootCode [ ]	オペレーティングシステムに関
するブートコード	

データ構造	
struct DirEntry	
{	
word	Status;
dword	SiblingPtr;
dword	PrimaryPtr;
dword	SecondaryPtr;
byte	Attributes;
word	Time;
word	Date;
byte	NameLen;
byte	Name [8];
byte	Ext [3];
}	
struct FileEntry	
{	
word	Status;
dword	SiblingPtr;
dword	PrimaryPtr;
dword	SecondaryPtr;
byte	Attributes;
word	Time;
word	Date;
byte	NameLen;
byte	Name [8];
byte	Ext [3];
}	
struct FileInfo	
{	
word	Status;
dword	ExtentPtr;
dword	PrimaryPtr;
dword	SecondaryPtr;
byte	Attributes;
word	Time;
word	Date;
word	ExtentLen;
word	UncompressedExtentLen;
}	
定義	
Name	ディレクトリ／ファイル名
Ext	ファイル拡張
Status	
bit #	
0	1 : レコードはディレクトリ構造内に
存在 (存在)	
0	1 : レコードはディレクトリ構造から削除済 (削除済)
1	1 : レコードは現属性、口付、及び時
50	間データを含む ( ATDRecent )

(18)

特開平7-191892

33

34

0 : レコードは置換されたデータを含むか、またはデータを含まない (ATDSuperseded)  
 3 - 2 1 1 : 未定義  
 1 0 : FileInfo  
 0 1 : FileEntry  
 0 0 : DirEntry  
 4 1 : PrimaryPtr は有効ではない  
 0 : PrimaryPtr は有効 (PrimaryPtrValid)  
 5 1 : SecondaryPtr は有効ではない  
 0 : SecondaryPtr は有効 (SecondaryPtrValid)  
 6 1 : SiblingPtr/ExtentPtr は有効ではない  
 0 : SiblingPtr/ExtentPtr は有効 (SiblingPtrValid/ExtentPtrValid)  
 DirEntry  
 15 - 7 保留  
 FileEntry  
 7 1 : ファイルは圧縮されていない  
 0 : ファイルは圧縮されている  
 15 - 8 圧縮アルゴリズムの識別  
 FileInfo  
 7 1 : ファイルは圧縮されていない  
 0 : ファイルは圧縮されている  
 8 1 : 範囲は圧縮されたブロックの最初のセグメントを含まない  
 0 : 範囲は圧縮されたブロックの最初のセグメントを含む  
 9 1 : 範囲は圧縮されたブロックの最後のセグメントを含まない  
 0 : 範囲は圧縮されたブロックの最後のセグメントを含む  
 15 - 10 保留  
 SiblingPtr 同胞連鎖内の次の DirEntry または FileEntryを指すポインタ  
 ExtentPtr FileInfoEntry に対応付けられた範囲を指すポインタ  
 PrimaryPtr DirEntry : ディレクトリ階層内の次に低いレベルの最初の DirEntry または FileEntry を指す  
 FileEntry : ファイルに対応付けられた FileInfo エントリのリンクされたりリストを指す  
 FileInfo : ファイルのための次の FileInfo エントリを指す  
 SecondaryPtr DirEntry : ディレクトリのための次の DirEntry エントリを指す ; SecondaryPtrを除くこのエントリの全内容は有効ではなく、無視される  
 FileEntry : ファイルのための次の FileEntryエントリを指す ; SecondaryPtrを除くこのエントリの全内容は有効ではなく、無視される  
 FileInfo : ファイルのための次の FileInfo エントリを

指す  
 Attributes 読出し専用、読出し/書き込み等のようなファイル属性  
 Time 創成または変更の時刻  
 Date 創成または変更の日付  
 NameLen バイトで表した名前及び拡張の長さ  
 Name [ 8 ] 名前  
 Ext [ 3 ] 拡張  
 10 ExtentLen バイトで表した範囲の長さ  
 本発明のファイルシステムは、ディレクトリ及びファイル構造をブロック消去可能な FEProm 内のブロック境界を横切って分散させる。ファイルシステムは、FEProm内の記憶装置の割当て及び割当て解除に FEProm 管理者を使用する。ファイルシステムは、上述したように、リンクされたりリスト内のポインタとしてハンドルを使用する。以下の説明では「ハンドル」及び「ポインタ」を互換可能に使用する。図 30 は、図 2 のディレクトリ階層の一部のブロック割当て例を示す。図 30 に示した部分はディレクトリ「ルート」、ディレクトリ「DOS」、ディレクトリ「ワード」、ファイル「AUTOEXEC.BAT」、及びファイル「COMMAND.COM」のための DirEntry 及び FileEntryレコードからなっている。ブロック 0 はディレクトリ「DOS」及びディレクトリ「ワード」を含み、ブロック 12 はディレクトリ「ルート」及びファイル「COMMAND.COM」を含み、そしてブロック 14 はファイル「AUTOEXEC.BAT」を含んでいる。  
 【0050】図 28 は、図 30 の例のブロック割当て構造及び領域例を示す。図 28 はブロック 0 2401、ブロック 12 2402、及びブロック 14 2403 を示す。ブロック 0 2401 は、領域 2420 及び 2430 を含む。領域 2420 はディレクトリ「DOS」のための DirEntry を含み、領域 2430 はディレクトリ「ワード」のための DirEntry を含む。ブロック 0 2401 は、見出し 2404、領域 2420 に対応する Alloc [ 0 ] エントリ 2421、及び領域 2430 に対応する Alloc [ 1 ] エントリ 2431 をも含む。ブロック 12 2402 は領域 2410 及び 2450 を含む。領域 2410 はディレクトリ「ルート」のための DirEntry を含み、領域 2450 はファイル「COMMAND.COM」のための FileEntryを含む。ブロック 12 2402 は、ブロック見出し 2405、領域 2410 に対応する Alloc [ 0 ] エントリ 2411、及び領域 2450 に対応する Alloc [ 1 ] エントリ 2451 をも含む。ブロック 14 2403 は、領域 2490 及び 2440 を含む。領域 2490 はブートレコードを含み、領域 2440 はファイル「AUTOEXEC.BAT」のための FileEntryを含む。ブロック 14 2403 は、ブロック見出し 2406、領域 2490 に対応する Alloc [ 0 ] エントリ 2491、及び領域 2440 に対応する Alloc [ 1 ] エントリ

(19)

特開平7-191892

35

2441をも含む。

【0051】図28では、ポインタ2407、2413、24233、2433、2443、2453、及び2493は、ポインタ2407から始まりブロック見出し2406内のブートレコードまでのディレクトリ階層を限定する。ブートレコード2490はブロック142403内にある。ブロック142403のための可変ステータスは、このブロックがブートディレクトリを含んでいることを指示する。BootRecordPtr2407はブートレコードのためのAlloc[0]エントリ2491を指し示している。Alloc[0]エントリ2491は、領域2490のオフセットを含む変数「オフセット」2492を含む。領域2490はブートレコードを含む。ブートレコードはディレクトリ「ルート」に対応するAlloc[0]エントリ2411を指し示すポインタRootDirectoryPtr2493を含む。Alloc[0]エントリ2411は、領域2410のオフセットを含む変数「オフセット」2412を含む。領域2410はディレクトリ「ルート」のためのDirEntryを含む。ディレクトリ「ルート」のPrimaryPtr2413はディレクトリ「DOS」に対応するAlloc[0]エントリ2421を指し示す。Alloc[0]エントリ2421は、領域2420のオフセットを含む変数「オフセット」2422を含む。

【0052】領域2420はディレクトリDOSのためのDirEntryを含む。ディレクトリ「DOS」のSiblingPtr2423はディレクトリ「ワード」のためのAlloc[1]エントリ2431を指し示す。Alloc[1]エントリ2431は、領域2430のオフセットを含む変数「オフセット」2432を含む。領域2430はディレクトリ「ワード」のためのDirEntryを含む。ディレクトリ「ワード」のSiblingPtr2433はファイル「AUTOEXEC.BAT」のためのAlloc[1]エントリ2441を指し示す。Alloc[1]エントリ2441は、領域2440のオフセットを含む変数「オフセット」2442を含む。領域2440はファイル「AUTOEXEC.BAT」のためのFileEntryを含む。ファイル「AUTOEXEC.BAT」のためのポインタSiblingPtr2443はファイル「COMMAND.COM」のためのAlloc[1]エントリ2451を指し示す。Alloc[1]エントリ2451は、領域2450のオフセットを含む変数「オフセット」2452を含む。領域2450はファイル「COMMAND.COM」のためのFileEntryを含む。SiblingPtr2453はFNULにセットされ、リンクされたリストの終わりを指示する。

【0053】このファイルシステムは、ディレクトリの追加及び削除、及びファイルの創成、拡張及び変更を可能にする。図7は、ディレクトリをFEPromへ追加するルーチンの流れ図である。このルーチンの入力パラメータは、新しいディレクトリの完全パスネームと、新しいディレクトリのための属性データである。このルーチンは、親ディレクトリのためのDirEntryのアドレスを含

36

むように変数Pをセットし、また子ディレクトリのためのDirEntryのアドレスを含むように変数Cをセットする。例えば、パスネーム“\P\C”は、ルートディレクトリのサブディレクトリであるPのサブディレクトリであるディレクトリCが創成されることを意味する。図8はディレクトリCがPの最初のサブディレクトリである場合を示し、図9はディレクトリCがPの最初のサブディレクトリではない場合を示す。図8及び9において、実線はディレクトリCを追加する前のディレクトリ構造を示し、点線はディレクトリCが追加された後のディレクトリ構造を示す。

【0054】図7のブロック401においてシステムは、ルートディレクトリからの経路を追跡することによってディレクトリPを探知し、ディレクトリPのためのDirEntryを指し示すように変数Pをセットする。ディレクトリPを探知する時システムは、変数SecondaryPtrによって置換されていない限り変数PrimaryPtrを追跡する。ブロック402では、システムはディレクトリCのためのDirEntryに領域を割当てる。システムは、FEProm管理者の手順を呼出すことによって領域を割当てる。システムは、割当て済領域を指し示すように変数Cをセットする。以下の説明ではFEProm管理者から戻されるハンドルを領域を指し示すポインタと呼ぶ。ブロック403においてシステムは、変数「名、時刻、日付、及び属性」を新たに割当てられたレコード内にセットし、新たに割当てられたエントリがディレクトリエントリであることを指示するように変数「ステータス」をセットする。

【0055】ブロック405乃至412においてシステムは、新しいディレクトリエントリを古いディレクトリ構造内へリンクする。システムは、ブロック406乃至410において新しいディレクトリがPの最初のサブディレクトリではない場合の状況処理し、ブロック411及び412において新しいディレクトリがPの最初のサブディレクトリである場合の状況処理する。ブロック405において、もしPrimaryPtrが有効であることをP→「ステータス」が指示すれば、ディレクトリPはサブディレクトリを有しているか、または有していたのであり、システムはブロック406へ進み、そうでない場合ディレクトリPはサブディレクトリを有していたことはなく、システムはブロック411へ進む。ブロック411では、システムは新たに割当てられたディレクトリエントリであるディレクトリCを指し示すようにP→PrimaryPtrをセットして新しいディレクトリへのリンクを遂行する。ブロック412においてシステムは、変数PrimaryPtrが有効であることを指示するようにP→「ステータス」をセットしてルーチンを終了する。

【0056】ブロック406においてシステムは、変数next-PtrをP→PrimaryPtrに等しくセットする。

(20)

特開平7-191892

37

変数 next-ptr は同胞サブディレクトリのリンクされたリスト内の次のディレクトリを指し示すポインタを含む。ブロック407において、SiblingPtrが有効であることを next-ptr によって指し示されるレコードのステータスが指示していればシステムはブロック408へ進み、そうでなければ同胞のリンクされたリストの終わりに到達したのであり、システムはブロック409へ進む。ブロック408では、next-ptr は next-ptr によって指し示されるレコードの SiblingPtr に等しくセットされ、これはリンクされたリスト内の次のディレクトリを指し示すように next-ptr を前進させ、システムはブロック407へ進められてリンクされたリストの終わりに達したか否かが決定される。同胞のリンクされたリストの終わりを探索する場合、システムは変数 SiblingPtr を追跡する。システムはブロック409において、next-ptr によって指し示されるレコードの SiblingPtr を、ディレクトリCのためのDirEntryを指し示すポインタに等しくセットする。ブロック410においてシステムは、新たに割当てられたディレクトリエントリを指し示すエントリ内の SiblingPtr が有効であることを指し示すように、next-ptr によって指し示されるレコードの「ステータス」をセットし、ルーチンを終了する。

【0057】図10は、新しいファイルに関して FileEntryレコードをファイルシステム内へ追加するルーチンの流れ図である。FileEntryレコードは階層的な木構造のファイルシステムの単なる葉ノードに過ぎず、FileEntryレコードを追加するルーチンは図7に示したディレクトリ追加ルーチンに酷似している。主要な相違点はブロック803において、レコードがファイルであることを指示するように変数「ステータス」がセットされることである。

【0058】図11及び12は、データをファイルの終わりに追加するためのルーチンの流れ図である。このルーチンには、完全パスネーム、書込まれるデータ及び書込まれるバイトの数が渡される。図13は拡張されるファイル「L.DAT」を含むディレクトリ構造のレイアウトの見本である。実線はファイルが拡張される前の構造を示し、点線はファイルが拡張された後の構造を示す。ファイル「L.DAT」は始めに FileEntryレコード1101、FileInfoレコード1102、及びそれに対応付けられた範囲1103を有している。点線は、範囲D2 1105内のファイルへ追加されるデータを有する FileInfoレコード1104を表す。

【0059】図11のブロック1001においてシステムは、領域を FEProm 内の新 FileInfo レコードに割当て、そのレコードを指し示すように変数 FI をセットする。システムはブロック1002において領域をデータ範囲に割当て、その範囲を指し示すように変数 D をセットする。システムはブロック1003において、割当てられたブロックへデータを書込む。ブロック1004に

38

おいてシステムは割当てられた FileInfo エントリ内に変数「属性、時刻、及び日付」をセットする。ブロック1005においてシステムは、FI → ExtentPtrを割当てられた範囲のハンドルにセットする。ブロック1005Aでは、FI → ExtentLenが範囲の長さを含むようにセットされる。ブロック1005Bでは FI → 「ステータス」が、「存在」、ATDRecent、FileInfo、及び ExtentPtrvalid にセットされる。ブロック1006ではシステムは、拡張すべきファイルの FileEntryレコードを探知し、そのレコードのアドレスに FE をセットする。好ましい実施例では、システムは新範囲及び FileInfo レコードを割当てる前に FileEntryレコードを探知して、何等かの割当てが行われる前にファイルが存在するようにしている。

【0060】図11の続きである図12のブロック1007乃至1012においてシステムは、拡張すべきファイルの最後の FileInfo レコード（もし1つが存在すれば）を探知する。システムは FileEntryレコード及び FileInfo レコードの PrimaryPtr または SecondaryPtr を追跡する。有効 SecondaryPtr は、PrimaryPtr によって指し示されるレコードが、SecondaryPtr によって指し示されるレコード内のデータによって置換されたことを指示する。ブロック1007においてシステムは、ポインタ next-ptr を FileEntryレコードを指し示すポインタに等しくセットする。ブロック1008Aでは、ポインタ prev-ptr が next-ptr に等しくセットされる。ファイル内の最後の FileInfo レコードが探知されると、ポインタ prev-ptr はそのレコードを指し示すポインタを含む。ブロック1009では、もし SecondaryPtr が有効であることを next-ptr によって指し示されるレコードのステータスが指示していれば、PrimaryPtr によって指し示されるレコード内のデータは置換されたのであり、システムはブロック1011へ進み、そうでない場合はシステムはブロック1010へ進む。ブロック1010においてシステムは next-ptr を、next-ptr によって指し示されるレコードの PrimaryPtr に等しくセットしてリンクされたリスト内の次のレコードを指し示すポインタを入手し、ブロック1012へ進む。ブロック1011ではシステムは next-ptr を、next-ptr によって指し示されるレコードの SecondaryPtr に等しくセットしてリンクされたリスト内の次のレコードを指し示すポインタを入手し、ブロック1008Aへ進む。ブロック1012では、もし next-ptr が有効であれば、リンクされたリストの終わりに達したのであり、システムはブロック1013へ進み、そうでない場合にはシステムは1008Aへ戻ってリンクされたリスト内の次のレコードを処理する。ブロック1013ではシステムは prev-ptr によって指し示されるレコードの PrimaryPtr を、FI を指し示すポインタに等しくセットしてファイルの拡張を遂行する。ブロック1014におい

(21)

特開平7-191892

39

てシステムはprev-Ptrによって指し示されるレコードの「ステータス」を PrimaryPtrValidに等しくセットしてルーチンを終了する。

【0061】図14及び15は、ファイル内のデータを更新する「ファイル更新」ルーチンの流れ図である。このルーチンのパラメタは、変更された関連範囲を持つためのFileInfoブロックのアドレスであるRと、新データのための範囲内へのオフセットである extent-offsetと、新データである new-data と、新データの長さである data-lengthとである。少なくともあるブロックが消去されるまでは、FEPromは1回書き込み装置であるから、データが記憶される領域は、ファイルへの更新が発生する時には再書き込みはできない。以下に説明するように好ましい実施例では更新されるデータは FEProm の異なる領域へ書き込まれる。

【0062】図16は、あるファイルのための FileInfo レコードのリンクされたリストの典型的部分を示す。ファイル更新ルーチンは陰影を施した領域1301によって表されるデータを置換する。図17は変更されるデータが FEProm へ書き込まれた後のリンクされたリストの構造を示す。3つの FileInfo レコードR1 1411、R2 1412、及びR3 1413がリンクされたリスト内へ挿入されている。範囲全体が再書き込みされるのではなく、実際に変更された部分だけが再書き込みされるのである。ルーチンは範囲を3つの区分D1 1401、D2 1402、及びD3 1403に分割する。区分D1 1401及びD3 1403は、更新によって変化しないデータを含み、区分D2 1402は変化するデータを含む。各区分は対応する FileInfo レコードを有する。FileInfo レコードR1 1411、R2 1412、及びR3 1413はそれらの PrimaryPtr フィールドを通してリンクされている。またR1 1411及びR3 1413内の ExtentPtr フィールドはそれらの対応範囲区分を指し示すようにセットされ、ExtentLen フィールドがセットされる。新範囲は、レコードR2 1412によって指し示される区分である新D2 1404に対応する新データに割当てられる。レコードR 1410の SecondaryPtr は FileInfo レコードR1 1411を指し示し、R 1410の PrimaryPtr が置換されたことを指示する。FileInfo レコードR3 1413の PrimaryPtr は FileInfo レコードR 1410の PrimaryPtr 内に含まれている値にセットされてリンクを完了する。ファイル更新ルーチンは、ブロック内に3つの新 Alloc エントリを追加するための範囲を含む十分なスペースが存在することに依存する。これら3つの Alloc エントリは、1つの領域ではなく3つの領域内に範囲を再定義する。もし十分なスペースが存在しなければ、ブロックの再利用によって空きスペースを発生させることができ、ファイル更新ルーチンを呼出すことができる。しかし、もし十分な空きスペースが存

40

在しなければ、その範囲内のデータは新範囲へ移動させられる。もしデータが移動させられれば、新データは古データと統合され、1つだけの FileInfo レコードを有する新ブロック内の1領域へ書き込まれる。古ブロック内の領域は割当て解除される。好ましい実施例では、FEProm 管理者は既存領域の部分の指し示すAlloc エントリの追加を支援する。図17の例は、ブロックへ追加する3つの新Alloc エントリを必要とし、これらはD1、D2、及びD3に関連付けられた新たに定義された領域に対応する。D2のための Alloc エントリは割当て解除され、D1及びD3のための Alloc エントリが割当てられる。区分D1、D2、及びD3からなる領域に対応する古い Alloc エントリのステータスは、それが置換されたことを指示するようにセットされる。置換済のステータスは、Alloc エントリが本質的に対応領域を用いずに割当て解除されることを指示する。

【0063】図14のブロック1201においてシステムは FileInfo レコードのための3つの領域を割当て、領域のアドレスを含むように変数R1、R2、及びR3をセットする。ブロック1202においてもしR→「ステータス」が ATDRecent を指示していれば、システムはR1→「時刻」、R1→「日付」、及びR1→「属性」をR内の値にセットし、R1→「ステータス」を ATDRecent にセットし、そうではない場合には、システムはこれらのフィールドを PNULL にしたままである。好ましい実施例では、FEProm 管理者は Alloc エントリを置換にセットするのと、Alloc エントリを既存領域に割当ててのを支援する。ブロック1203においてシステムはある領域を新データに割当て、R2NewDataをその領域のアドレスにセットする。ブロック1204では、3つの Alloc エントリが割当てられる。これらのエントリはD1、D2、及びD3を指し示すように初期化される。D1、D2、及びD3からなる領域を指し示す Alloc エントリのステータスは置換にセットされる。ブロック1205においてシステムは R2NewDataによってアドレスされた新領域へ new-data を書き込む。システムはブロック1206乃至1208Aにおいてデータを FileInfo レコードR2内にセットする。ブロック1206においてシステムは、R2→ExtentPtr を新データのための領域を指し示すポインタに等しくセットする。ブロック1207では、R2→ExtentLen を新領域の長さに等しくセットする。ブロック1208において、R2→PrimaryPtr がR3を指し示すポインタにセットされる。ブロック1208Aにおいてシステムは、ExtentPtr 及び PrimaryPtr が有効であることを指示するようにR2→「ステータス」をセットする。

【0064】図14の続きである図15のブロック1209乃至1211Aにおいてシステムは、データを FileInfo レコードR3内にセットする。ブロック1209ではD3領域を指し示すポインタに等しくR3→Ext

(22)

特開平7-191892

41

entPtrをセットする。ブロック1210ではR3→ExtentLenをD3領域の長さに等しくセットする。ブロック1211においてシステムは、R3→PrimaryPtrをR→PrimaryPtrに等しくセットする。ブロック1211Aでは、ExtentPtr及びPrimaryPtrが有効であることを指示するようにR3→「ステータス」をセットする。

【0065】ブロック1212乃至1214Aにおいてシステムは、データをFileInfoレコードR1内にセットする。ブロック1212ではD1領域を指し示すポインタに等しくR1→ExtentPtrをセットする。ブロック1213ではR1→ExtentLenをD3領域の長さに等しくセットする。ブロック1214においてシステムは、R1→PrimaryPtrをR2を指し示すポインタにセットする。ブロック1214Aでは、ExtentPtr及びPrimaryPtrが有効であることを指示するようにR1→「ステータス」をセットする。

【0066】ブロック1215においては、R1を指し示すポインタに等しくなるようにR→SecondaryPtrがセットされる。ブロック1216では、SecondaryPtrが有効であることを指示するようにR→「ステータス」がセットされる。これでルーチンが終了する。図18及び19は、ファイル更新の特別な場合の結合のFileInfoリストを示す。これらの特別な場合を処理するためのルーチンは、図14、15に示したファイル更新の一般的な場合の処理に必要なルーチンの部分集合である。図18において、範囲の始まりから始まるデータが更新される。区分D1 1501は更新すべき範囲の始まりのデータを含み、区分D2 1502は更新されない範囲の終わりのデータを含む。2つの新FileInfoレコードだけが必要である。第1のFileInfoレコードR1 1511は新データ1503を指し示し、第2のFileInfoレコードR2 1512は区分D2 1502内の古いデータを指し示す。図19に示すように範囲境界上で終わるデータが更新される場合にも同様な状況が発生する。ファイル更新の一般的な場合のように、D1及びD2を含む古い領域は、古い領域を含むブロックに2つの新しい割当て表エントリを割当てることによって2つの領域に細分される。またもしエントリのために十分なスペースが存在しなければ、変更されないデータは新ブロックへ移動させられる。

【0067】図20は更新データが範囲境界にまたがる場合のFileInfoレコードのためのリンクされたリストを示す。図21は、FEPromからディレクトリを削除するルーチンの流れ図である。ファイルを削除するルーチンは、対応付けられたFileInfoレコードが割当て解除されること以外は同一である。このルーチンは、DirEntryのステータスを、それが削除されることを指示するようにセットする。ブロック1801においてシステムは、削除すべきディレクトリを探知し、そのディレクト

42

リアドレスを含むように変数「ポインタD」をセットする。ブロック1802では、そのディレクトリを削除することを指示するようにD→「ステータス」をセットする。

【0068】ディレクトリまたはファイルの名前は、新しいDirEntryまたはFileEntryをそれぞれ割当て、次いで新しいエントリを指し示すように古いエントリのSecondaryPtrをセットすることによって変更される。図23は、名前が“B.DAT”に変更される時の、“B.DAT”のためのファイルエントリを実線で、また変更を点線で示してある。新エントリはFileInfoエントリ、ディレクトリ構造、及び古いエントリに対応付けられた範囲のリンクされたリストを指し示す。

【0069】図22はファイル名の変更を実現する好ましいサブルーチンの流れ図である。ディレクトリ名を変更するためのサブルーチンは、対応付けられる範囲が存在しないことを除いて同一である。このルーチンへの入力パラメタは、ファイルの同胞及び新ファイル名である。ブロック1901においてシステムはディレクトリを通してシステムを探索し、名前を変更すべきファイルを探知し、FileEntryを指し示すように変数Pをセットする。システムは、そのファイルのためのエントリのリンクされたリスト内の最後のFileEntryを探索する。ファイルは各名前変更毎にエントリを有する。

【0070】ブロック1904においてシステムは、領域を新FileEntryに割当て、その領域を指し示すように変数Cをセットする。ブロック1905においてシステムはC→「名前」を新ファイル名に等しくセットし、C→「属性」、C→「日付」、C→「時刻」をセットし、ATDRecentに置換されるファイルエントリに基づいてC→「ステータス」をセットする。ブロック1906においてシステムは、C→SiblingPtrをP→SiblingPtrに等しくセットしてエントリをディレクトリ階層にリンクする。ブロック1909では、C→PrimaryPtrがP→PrimaryPtrに等しくセットされ、新エントリは範囲のリストにリンクされる。ブロック1909Aでは、ExtentPtr及びPrimaryPtrが有効であることを指示するためにC→「ステータス」がセットされる。ブロック1910ではシステムはCを指し示すポインタに等しくP→SecondaryPtrをセットする。ブロック1911においてシステムはSecondaryPtrが有効であることを指示するようにP→「ステータス」をセットし、古いエントリの置換を完了させてルーチンを終了させる。

【0071】図24及び25は、あるファイルの属性データを変更するルーチンの流れ図である。あるファイルに関連する属性データは、ATDRecentのステータスを有し、また「性、日付、及び時刻」フィールドがFNULLにセットされている第1のFileInfoエントリを選択することによって変更される。もしこのようなフィールドが

(23)

特開平7-191892

43

存在しなければ、新しい FileInfo エントリが作成され、選択される。次いでシステムは「属性、日付、及び時刻」フィールドを選択された FileInfo エントリ内にセットする。先に最新の「属性、日付、及び時刻」データを記憶していた FileInfo エントリは、 ATDSuperseded にセットされているステータスを有している。入力パラメタはパスネーム及び属性データである。ブロック 2101 においてシステムはディレクトリ構造を通して探索してファイルを探知し、 FileEntry を指し示すように変数 P をセットする。ブロック 2102 においてもし P → 「ステータス」が ATDRecent を指示していれば、FileEntry は最新の属性データを含んでおり、システムはブロック 2103 に進み、そうでない場合にはシステムはブロック 2104 に進む。ブロック 2103 においてシステムは変数 X を変数 P にセットして図 25 のブロック 2111 に進む。ブロック 2104 ではシステムは変数 C を P → PrimaryPtr に等しくセットする。システムはブロック 2105 乃至 2108 をループして最新の属性データを指示しているステータスを有する FileInfo エントリを探索する。ブロック 2105 においてもし SecondaryPtr が有効であることを C → 「ステータス」が指示していれば、システムはブロック 2106 へ進み、そうでない場合はシステムはブロック 2107 へ進む。ブロック 2106 では変数 C は C → SecondaryPtr にセットされて置換されるエントリを指し示すようにされ、ブロック 2105 へループバックされる。ブロック 2107 においてもし C → 「ステータス」が ATDRecent を指示していれば FileInfo エントリは最新の属性データを含んでいるのであり、システムはブロック 2109 へ進み、そうでない場合にはシステムはブロック 2108 へ進む。ブロック 2108 においてシステムは変数 C を C → PrimaryPtr に等しくセットし、ブロック 2105 へループバックする。ブロック 2109 においてはシステムは変数 X を変数 C にセットし、図 25 のブロック 2111 へ進む。

【0072】システムは、ブロック 2111 において変数 Y を変数 X に初期化する。変数 X は最新の属性データを有するエントリを指し示す。ブロック 2112 乃至 2119 においては、再新のステータスを有し FNUL にセットされた属性データを有する次のエントリを探知する。ブロック 2112 においてもし PrimaryPtr が有効であることを Y → 「ステータス」が指示していれば、システムはブロック 2113 へ進み、そうでなければ新エントリを追加するためにブロック 2120 へ進む。ブロック 2113 では、変数 Y は Y → PrimaryPtr にセットされる。ブロック 2114 においてもし SecondaryPtr が有効であることを Y → 「ステータス」が指示していれば、システムはブロック 2115 へ進み、そうでなければブロック 2116 へ進む。ブロック 2115 では変数 Y が Y → SecondaryPtr に等しくセットされ、

44

ブロック 2114 へループバックされる。ブロック 2116 においてもし Y → 「ステータス」が ATDRecent にセットされていればシステムはブロック 2117 へ進み、そうでなければブロック 2112 へ戻される。ブロック 2117 においてもし Y → 「属性」、Y → 「日付」、及び Y → 「時刻」が FNUL に等しければシステムはブロック 2118 へ進み、そうでなければブロック 2112 へループバックされる。ブロック 2118 において、Y → 「属性」、Y → 「日付」及び Y → 「時刻」は新しい属性データにセットされる。ブロック 2119 では X → 「ステータス」が ATDSuperseded に等しくセットされ、ルーチンは完了する。

【0073】ブロック 2120 乃至 2123 においてシステムは新 FileInfo エントリを割当てて初期化する。ブロック 2120 では新 FileInfo エントリが割当てられ、新エントリを指し示すように変数 Z がセットされる。ブロック 2121 では、システムは Z → 「属性」、Z → 「日付」、及び Z → 「時刻」を新属性データにセットし、Z → 「ステータス」を「存在」、ATDRecent、及び FileEntry にセットし、Z → ExtentPtr を Y → ExtentPtr にセットし、そして Z → ExtentLen を Y → ExtentLen にセットする。ブロック 2122 では Y → SecondaryPtr が変数 Z に等しくセットされ、SecondaryPtr が有効であることを指示するように Y → 「ステータス」がセットされる。ブロック 2123 では、X → 「ステータス」が ATDSuperseded に等しくセットされ、エントリが最早現属性データを含んでいないことを指示し、ルーチンが完了する。

【0074】以上に本発明を好ましい実施例に関して説明したが、本発明はこの実施例に限定されるものではない。当業者ならば本発明の思想に基づく多くの変更が明白であろう。本発明の範囲は特許請求の範囲によってのみ限定されるものである。

【図面の簡単な説明】

【図 1】ディレクトリ及びファイルのサンプルハイアラキ構造もしくはツリー構造の編成を示す図である。

【図 2】図 1 の同じディレクトリ構造を示すリンクされたリスト構造体の図である。

【図 3】図 1 の同じディレクトリ構造を表す別のリンクされたリスト構造体を示した図である。

【図 4】ファイル名「\A\d. DAT」のファイルに対するリンクされたリストの構造を示す図である。

【図 5】ファイル名「\A\d. DAT」のファイルに対する別のリンクされたリストの構造を示す図である。

【図 6】本発明の好ましい実施例によるブロック消去可能な FE Prom のレイアウトを示す図である。

【図 7】本発明の好ましい実施例による Add Directory ルーチンを示す流れ線図である。

【図 8】本発明の好ましい実施例によりディレクトリ構造に新たにディレクトリが追加される前後の図である。



(24)

特開平7-191892

45

【図9】本発明の好ましい実施例によりディレクトリ構造に新たにディレクトリが追加される前後の図である。

【図10】本発明の好ましい実施例によるAdd\_\_Fileルーチンを示す流れ線図である。

【図11】本発明の好ましい実施例によるExtend\_\_Fileルーチンを示す流れ線図である。

【図12】本発明の好ましい実施例によるExtend\_\_Fileルーチンを示す流れ線図である。

【図13】本発明の好ましい実施例を用いたサンプルディレクトリ及びファイルレイアウトを示す図である。

【図14】本発明の好ましい実施例によるUpdate\_\_Fileルーチンを示す流れ線図である。

【図15】本発明の好ましい実施例によるUpdate\_\_Fileルーチンを示す流れ線図である。

【図16】本発明の好ましい実施例により更新される前後のファイルのサンプル部分を示す図である。

【図17】本発明の好ましい実施例により更新される前後のファイルのサンプル部分を示す図である。

【図18】本発明の好ましい実施例によりファイルが更新された後のファイルのサンプル部分を示す図である。

【図19】本発明の好ましい実施例によりファイルが更新された後のファイルのサンプル部分を示す図である。

【図20】本発明の好ましい実施例によりファイルが更新された後のファイルのサンプル部分を示す図である。

【図21】本発明の好ましい実施例によるDel\_\_Directoryルーチンを示す流れ線図である。

【図22】本発明の好ましい実施例によるChange\_\_File\_\_Nameルーチンを示す流れ線図である。

【図23】名前を変化したファイルに対するディレクトリ構造の前後を表す図である。

【図24】本発明の好ましい実施例によるChange\_\_Attribute\_\_Dataルーチンを示す流れ線図である。

【図25】本発明の好ましい実施例によるChange\_\_Attribute\_\_Dataルーチンを示す流れ線図である。

【図26】本発明の好ましい実施例におけるブロックのレイアウトを示す図である。

【図27】好ましい実施例においてリクレーミングを行った後の図26のブロックのレイアウトを示す図である。

46

【図28】図30のサンプルに対するサンプルブロック割り当て構造及び領域を示す図である。

【図29】ハンドルの参照解除を示す図である。

【図30】図2のディレクトリハイアラキの一部に対するサンプルブロック割り当てを示す図である。

【図31】好ましい実施例における初期化プロセスを示す流れ線図である。

【図32】好ましい実施例におけるブロック割り当てルーチンを示す流れ線図である。

【図33】好ましい実施例における領域割り当てルーチンを示す流れ線図である。

【図34】好ましい実施例におけるブロックリクレーミングルーチンを示す流れ線図である。

【図35】All locエントリのステータスを示す状態図である。

【図36】ブロックのステータスを示す状態図である。

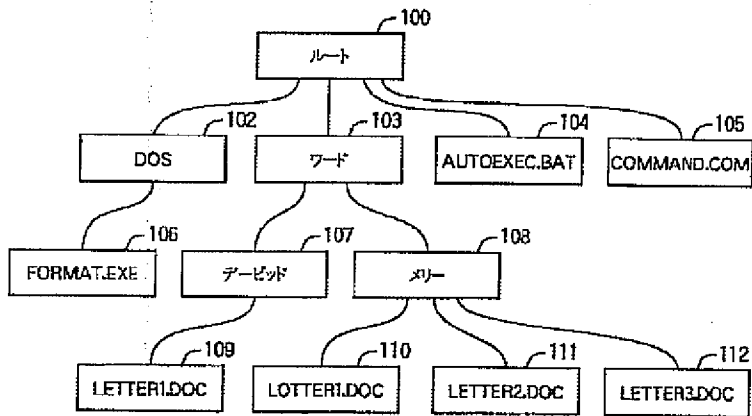
【符号の説明】

- 100 ディレクトリ「ルート」
- 102 「DOS」サブディレクトリ
- 103 「ワード」サブディレクトリ
- 104 「AUTOEXEC. BAT」ファイル
- 105 「COMMAND. COM」ファイル
- 106 「FORMAT. EXE」ファイル
- 107 「デービッド」サブディレクトリ
- 108 「メリー」サブディレクトリ
- 109 「LETTER1. DOC」ファイル
- 110 「LETTER1. DOC」ファイル
- 111 「LETTER2. DOC」ファイル
- 112 「LETTER3. DOC」ファイル
- 113 「ビル」ディレクトリ
- 120-132 ポインタ
- 140-142 リンクされたリスト
- 200-202 ファイル
- 203-206 範囲レコード
- 211-214 範囲
- 301 FEProm
- 302 ブロック
- 2302 ブロック割り当て構造体
- 2303 データ領域
- 2304 自由スペース
- 2310-2315 オフセット

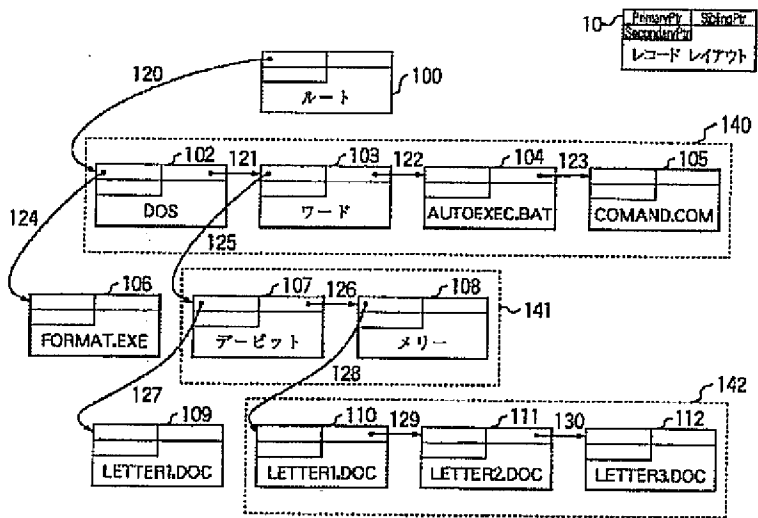
(25)

特開平 7-191892

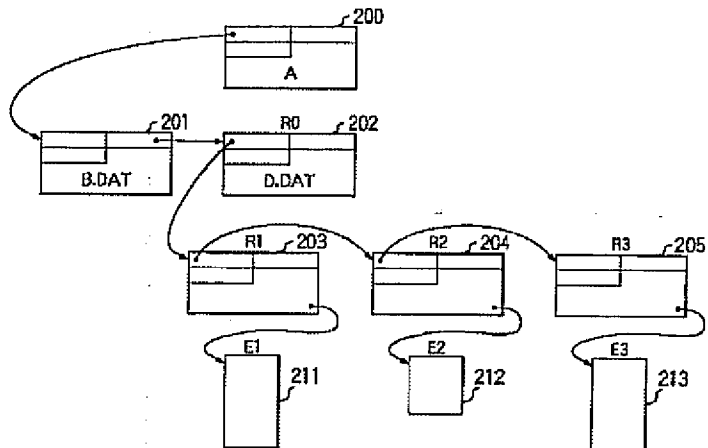
【図 1】



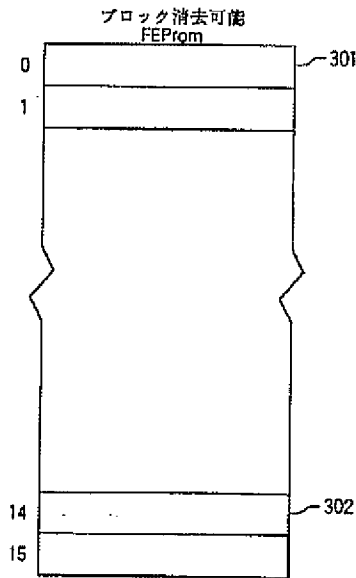
【図 2】



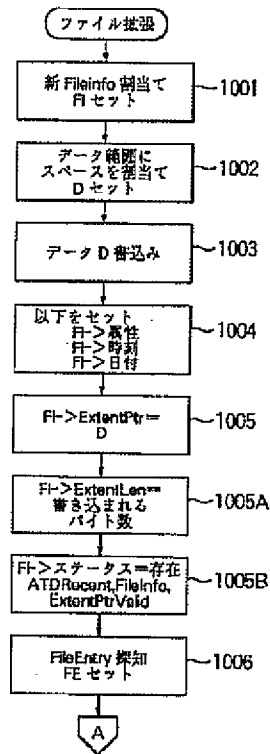
【図 4】



【図 6】



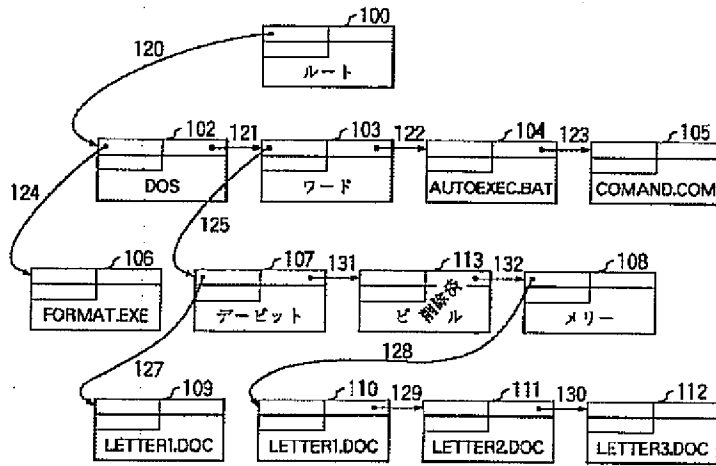
【図 11】



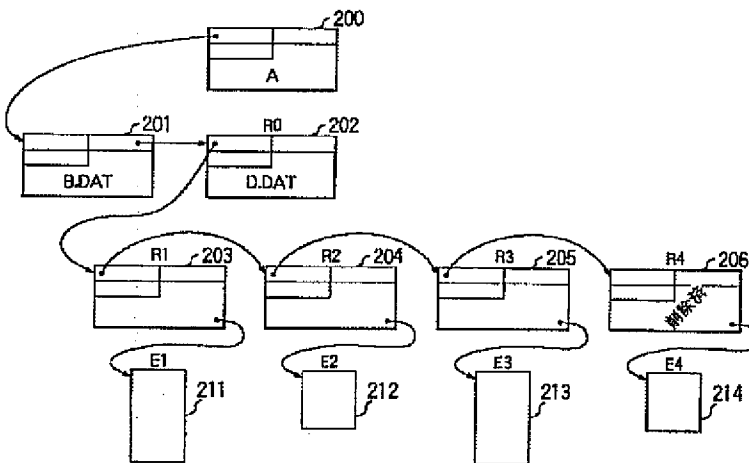
(26)

特開平7-191892

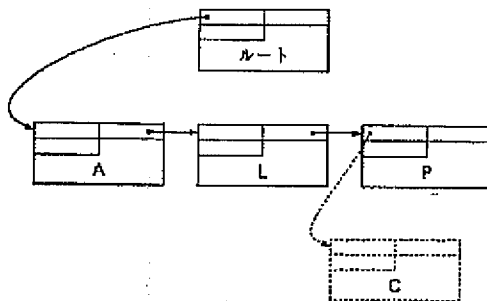
【図3】



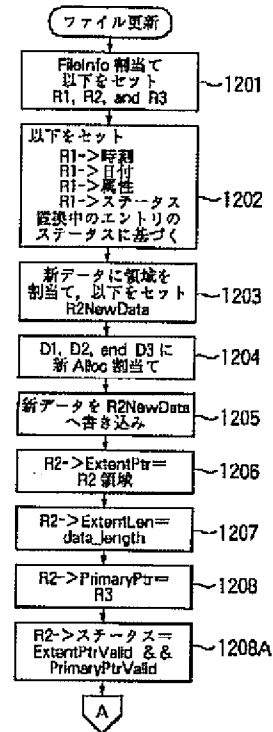
【図5】



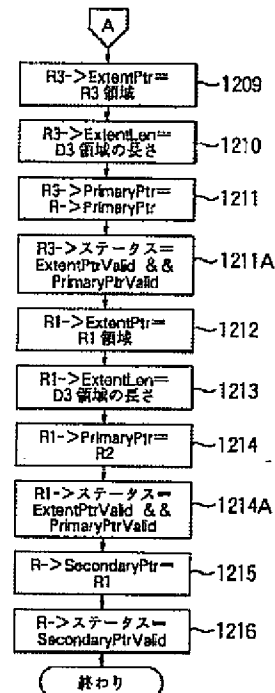
【図8】



【図14】



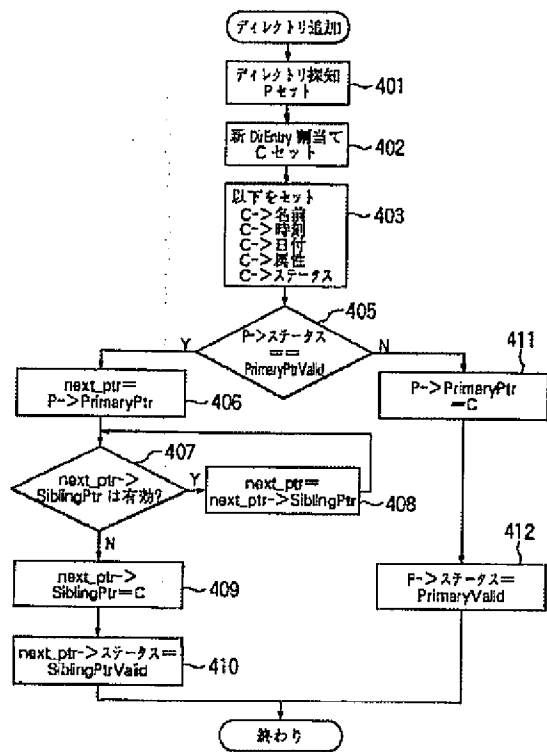
【図15】



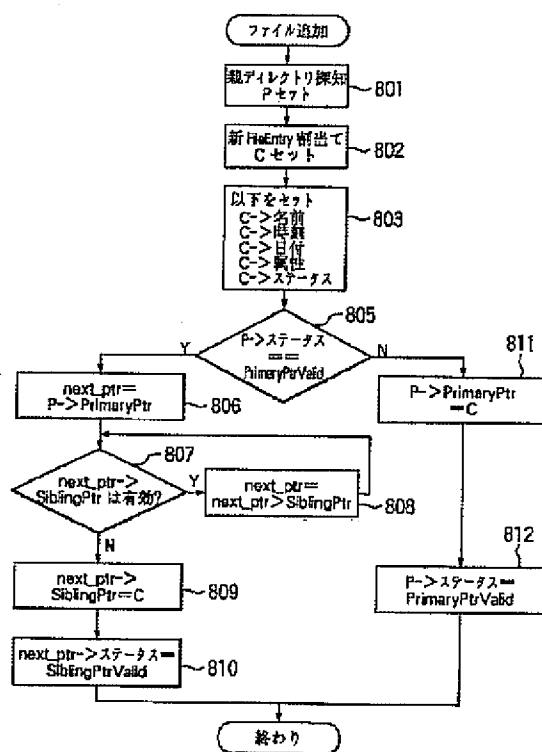
(27)

特開平 7-191892

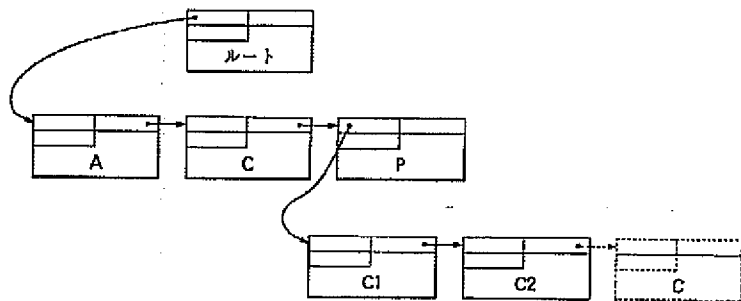
【図 7】



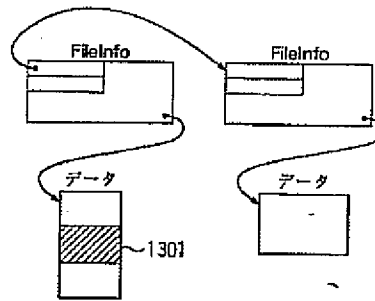
【図 10】



【図 9】



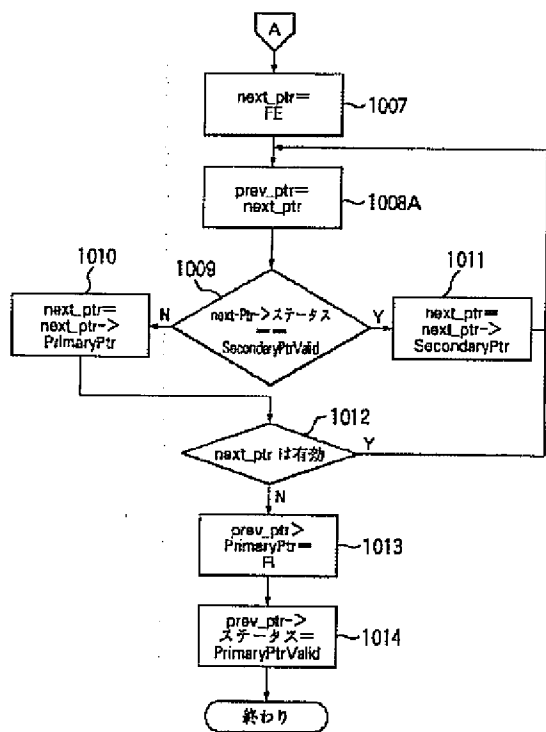
【図 16】



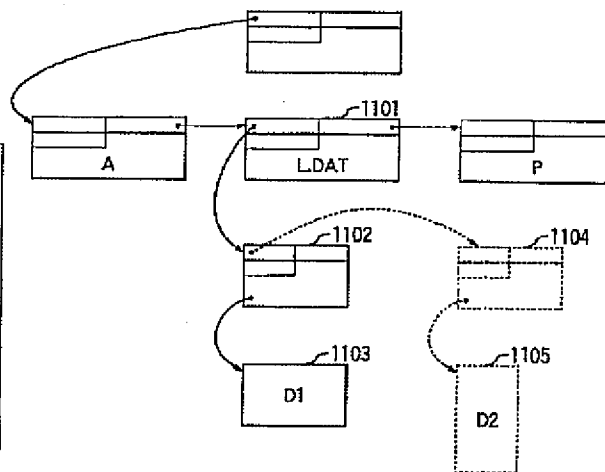
(28)

特開平7-191892

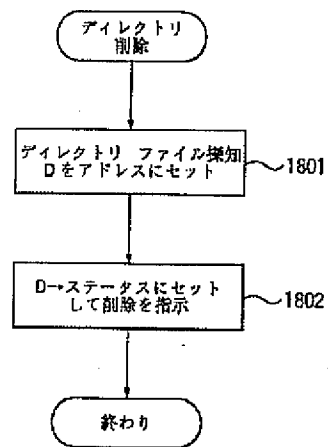
【図12】



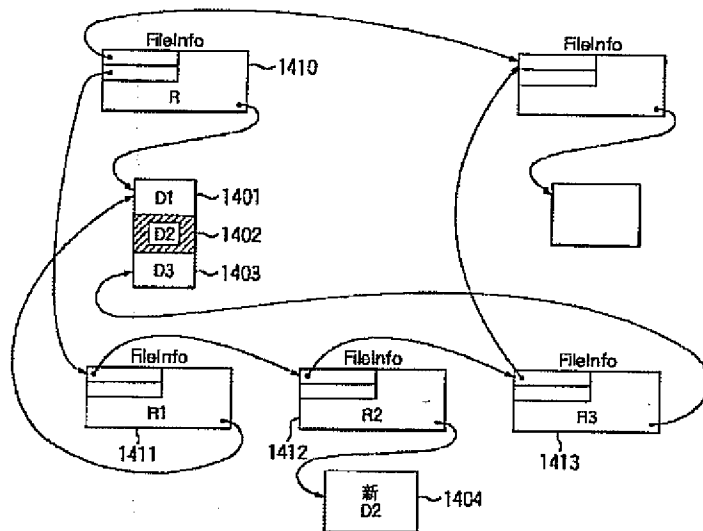
【図13】



【図21】



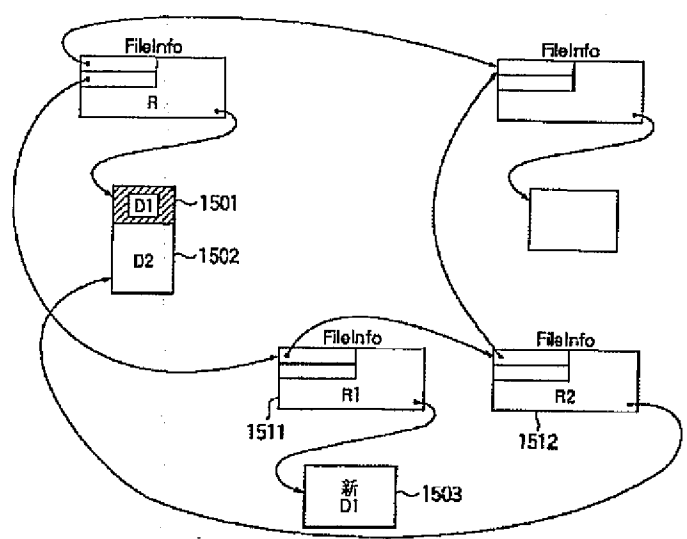
【図17】



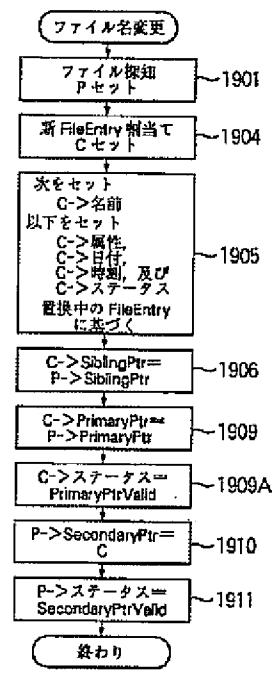
(29)

特開平7-191892

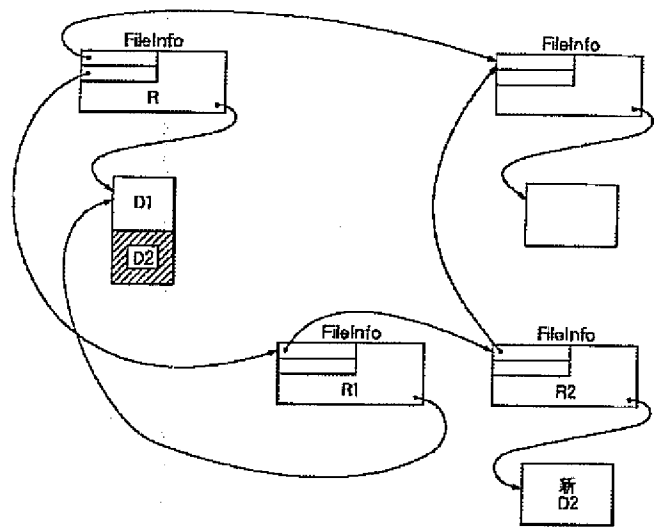
【図18】



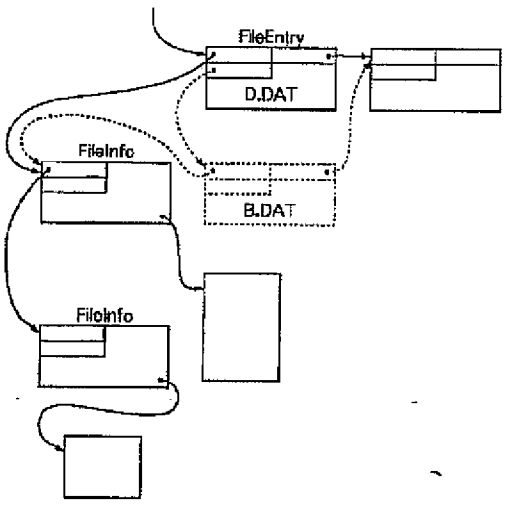
【図22】



【図19】



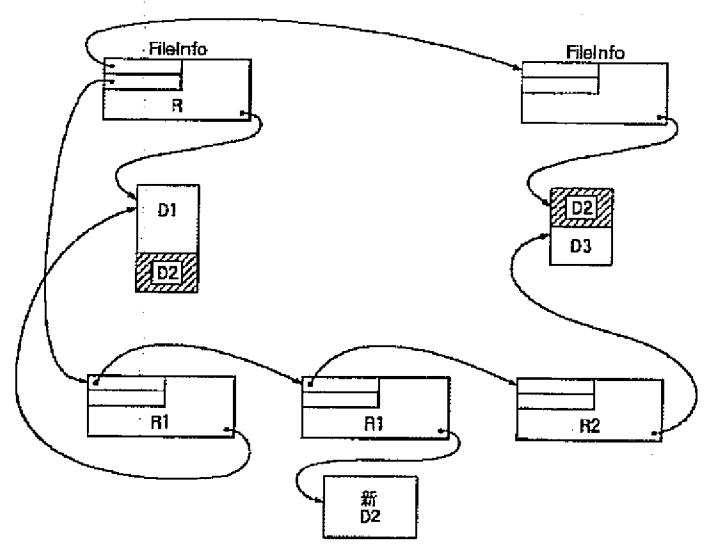
【図23】



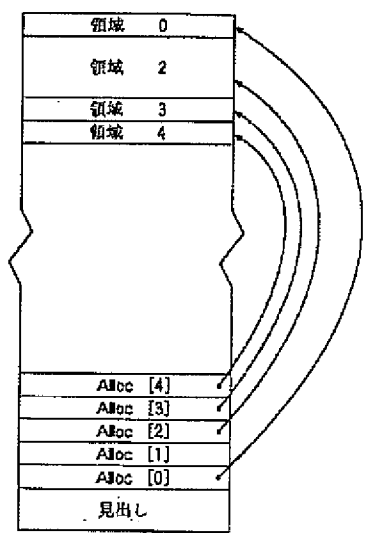
(30)

特開平7-191892

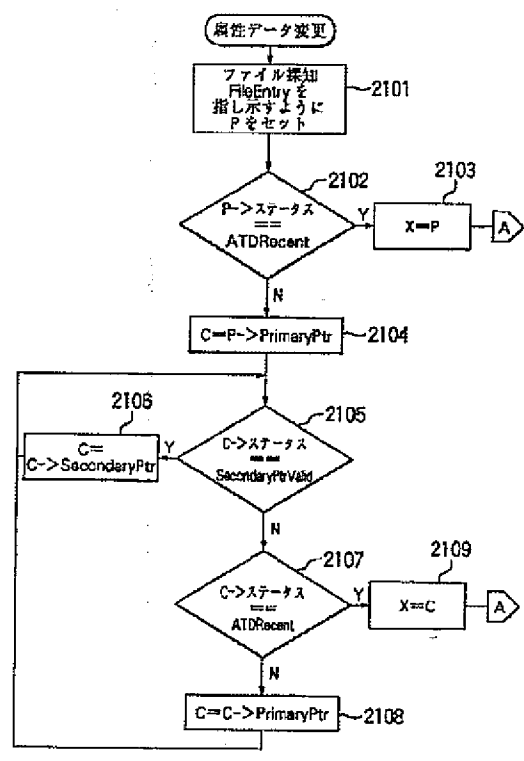
【図20】



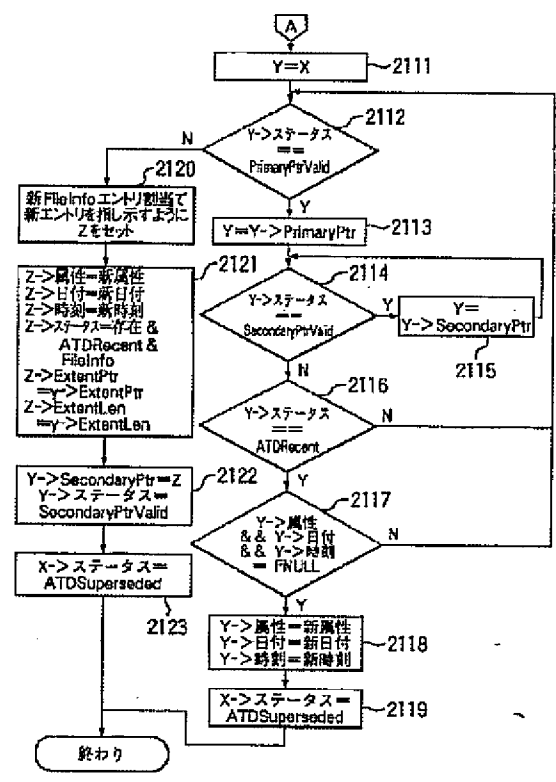
【図27】



【図24】



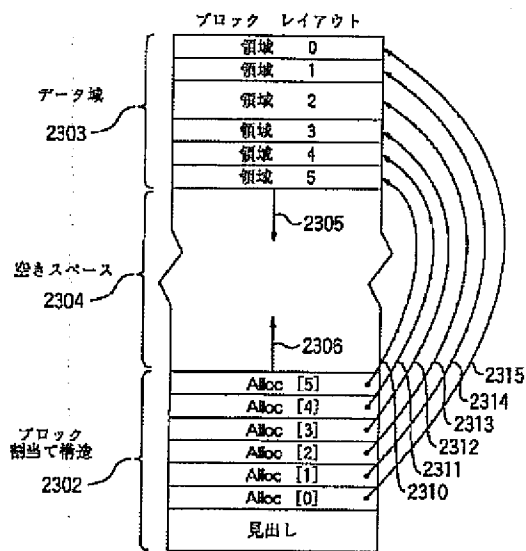
【図25】



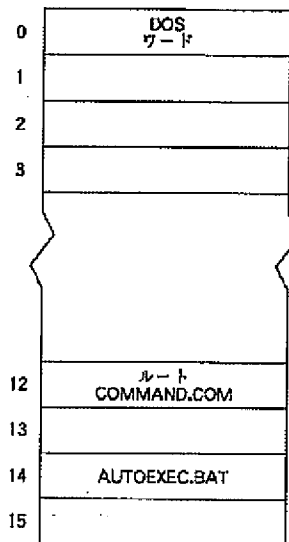
(31)

特開平7-191892

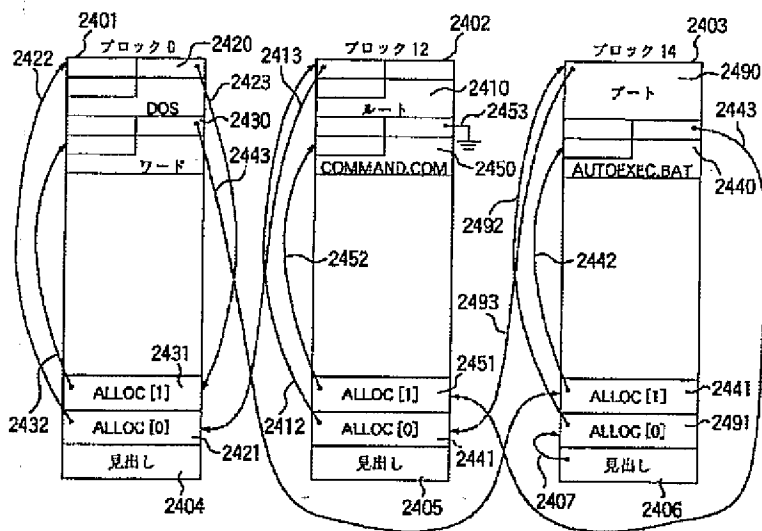
【図26】



【図30】



【図28】

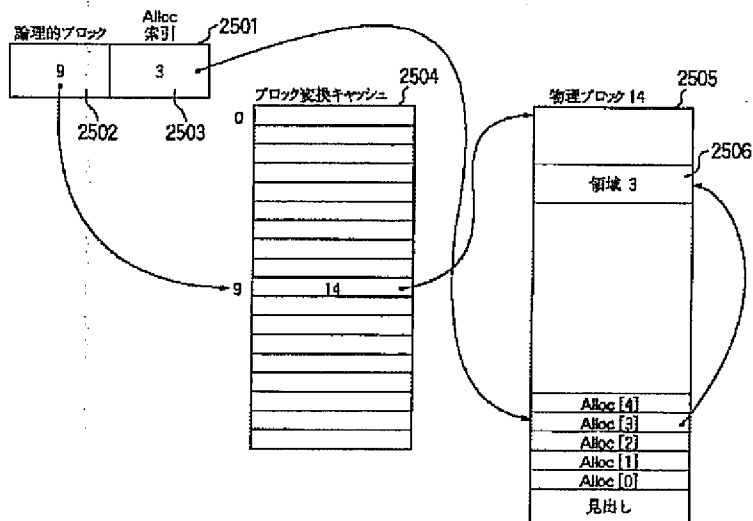




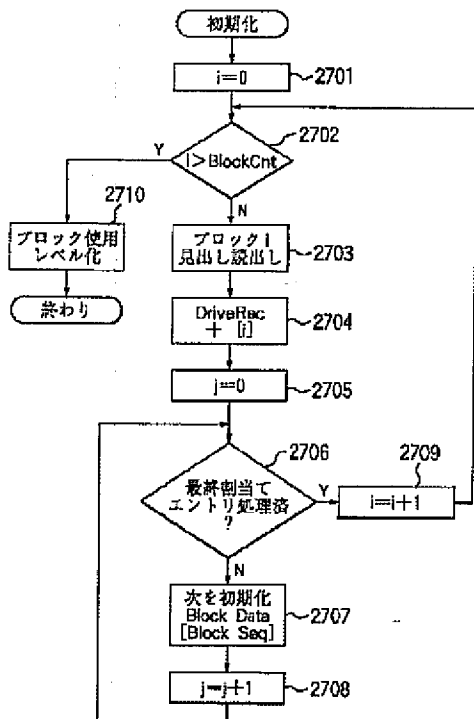
(32)

特開平7-191892

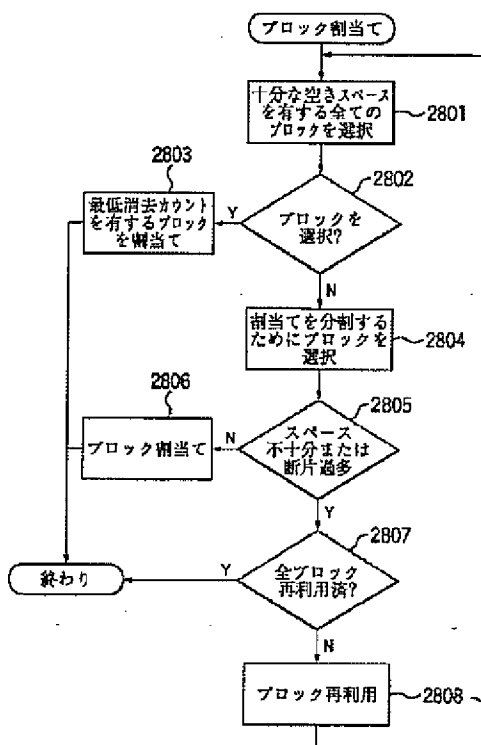
【図29】



【図31】



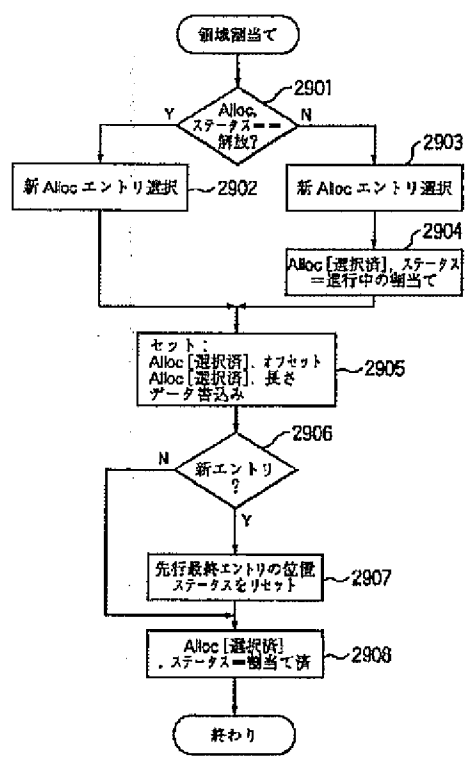
【図32】



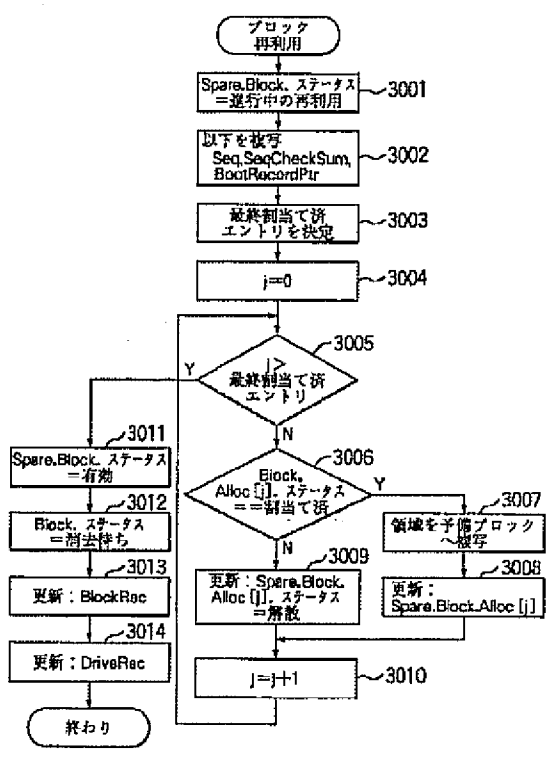
(33)

特開平7-191892

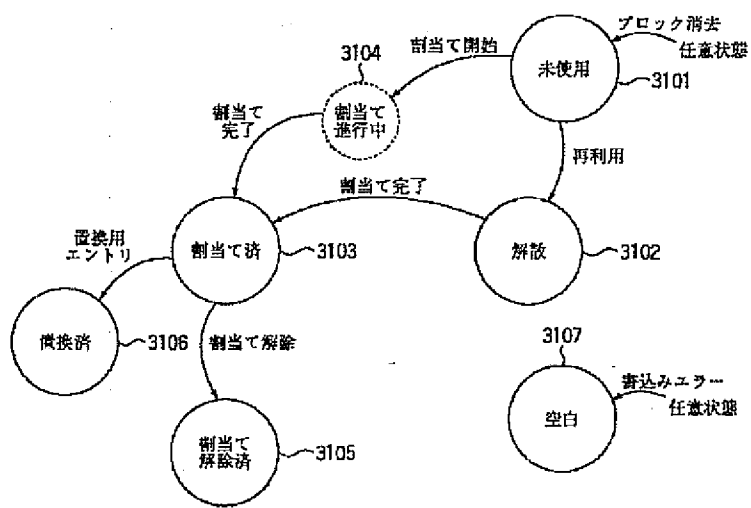
【図33】



【図34】



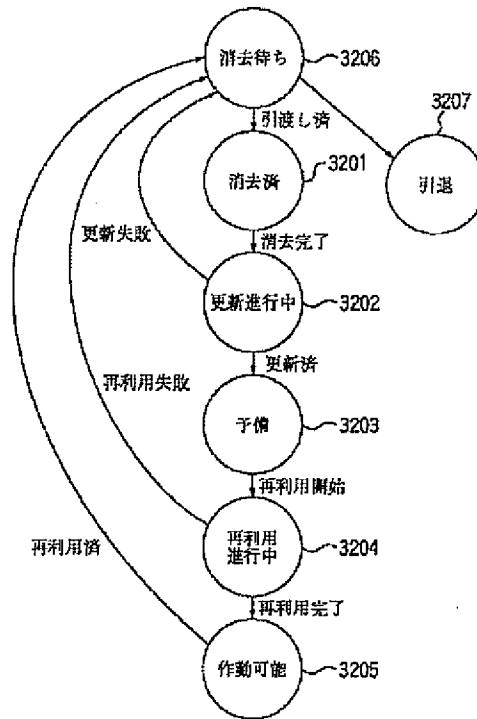
【図35】



(34)

特開平 7-191892

【図 36】



フロントページの続き

(72)発明者 スリラム ラジャゴパラン  
 アメリカ合衆国 ワシントン州 98007  
 ベルビュー ビー201 ノースイースト  
 サーフティフィフス 14550